Algorithms for Protein Structure Prediction

Paluszewski, Martin

Publication date:
2008

Document version
Publisher's PDF, also known as Version of record

Citation for published version (APA):
Paluszewski, M. (2008). *Algorithms for Protein Structure Prediction*. Department of Computer Science, University of Copenhagen.

Department of Computer Science
University of Copenhagen

Algorithms for Protein Structure Prediction

Ph.D. Thesis

Martin Paluszewski

September 2008

Acknowledgements

My acknowledgements go to all the people who I have been working with during this Ph.D.-study. This is especially my supervisor, Associate Professor Pawel Winter who also encouraged me to begin my Ph.D.-study. Special thanks go to Professor Kevin Karplus and the rest of his lab for making our collaboration in Santa Cruz possible. I would like to thank my research collaborators and paper co-authors Thomas Hamelryck and Rasmus Fonseca for valuable contributions and discussions. With the background in computer science, my knowledge of proteins was minimal before I began this study. Fortunately, my lovely wife Hanne Hammerbak has an educational background in biology and could answer most of my questions involving protein biology. Our wonderful sons Bjarke and Toke are my biggest motivation for doing research that hopefully will have a positive effect in their time.

Abstract

The problem of predicting the three-dimensional structure of a protein given its amino acid sequence is one of the most important *open* problems in bioinformatics. One of the carbon atoms in amino acids is the C_α-atom and the overall structure of a protein is often represented by a so-called C_α-trace.

Here we present three different approaches for reconstruction of C_α-traces from predictable measures. In our first approach [63, 62], the C_α-trace is positioned on a lattice and a tabu-search algorithm is applied to find minimum energy structures. The energy function is based on half-sphere-exposure (HSE) and contact number (CN) measures only. We show that the HSE measure is much more information-rich than CN, nevertheless, HSE does not appear to provide enough information to reconstruct the C_α-traces of real-sized proteins. Our experiments also show that using tabu search (with our novel tabu definition) is more robust than standard Monte Carlo search.

In the second approach for reconstruction of C_α-traces, an exact branch and bound algorithm has been developed [67, 65]. The model is discrete and makes use of secondary structure predictions, HSE, CN and radius of gyration. We show how to compute good lower bounds for partial structures very fast. Using these lower bounds, we are able to find global minimum structures in a huge conformational space in reasonable time. We show that many of these global minimum structures are of good quality compared to the native structure. Our branch and bound algorithm is competitive in quality and speed with other state-of-the-art decoy generation algorithms.

Our third C_α-trace reconstruction approach is based on bee-colony optimization [24]. We demonstrate why this algorithm has some important properties that makes it suitable for protein structure prediction.

Our approach for model quality assessment (MQA) [64] makes use of distance constraints extracted from alignments to templates. We show how to use CN probabilities in an optimization algorithm for selecting good distance constraints and we introduce the concept of non-contacts. When comparing our algorithm with state-of-the-art MQA algorithms on the CASP7 benchmark, our algorithm is among the top-ranked algorithms. We are currently participating in CASP8 MQA with this algorithm.

Preface

I began my Ph.D.-studies in June, 2005 at the Department of Computer Science, University of Copenhagen under supervision of Associate Professor Pawel Winter. At that time our algorithms and optimization group had much experience with network problems, production planning, packing problems and transportation problems, but little experience with computational biology. We therefore decided, to begin attacking problems in computational biology using our expertise for solving complex optimization problems. One of the main purposes of my Ph.D.-study therefore was to identify suitable problems in the field of protein structure prediction and get our group involved in the field. In 2007 I was working together with professor Kevin Karplus for 6 months. He is the head of the protein structure prediction group at the University of California, Santa Cruz (UCSC). During that period, I learned much about many of the problems that exist in the field of protein structure prediction and I was introduced to the field of protein model quality assessment.

General Outline

This thesis summarizes the research I have been involved with during my Ph.D.-study. It consists of 5 papers and 2 posters together with a text describing the background of our work.

The most important parts of this Ph.D.-thesis are the papers included in the appendices (pages 107 – 193). They contain detailed descriptions of the algorithms we have developed and all our research results. For people in the field of protein structure prediction, no prerequisites for reading the papers should be needed. As is the case with most scientific papers, they are written *by* experts in the field *to* other experts in the field. The introduction and background text here (pages 9 to 105) is therefore aimed at scientists and students with little or no background in bioinformatics. I therefore also allow myself to be less formal in this text. It does not contain the details of our research and should therefore be weighted less than the papers in the evaluation of this thesis.

Most chapters contain an introductionary description of the topic and a few illustrative examples of important results in the literature. The chapters also contain sections called *Our Research*, which describe how we apply the concepts and results of the given chapter in our research.

Contents

The following background and introduction text, papers and posters constitute this thesis:

Contents

Chapter 1

Introduction

A protein is a complex molecule consisting of thousands of atoms that interact with each other and with surrounding molecules. Proteins are very important molecules in all living organisms and are often referred to as *the molecules of life*. Knowing the native structure of proteins is fundamental for our understanding of their functionality, since protein structure directly determines protein function. Today, the three-dimensional structures of proteins are found using X-ray crystallography or *nuclear magnetic resonance* (NMR) experiments. These methods are quite expensive and they can be very time consuming. Furthermore, these methods cannot be be applied to all proteins, especially many of the membrane proteins [93, 59].

1.1 Protein Structure Prediction is an Open Problem

One of the most important *open* problems in bioinformatics is therefore to find an algorithm that takes an amino acid sequence as input and outputs the native structure of the protein (i.e., the three-dimensional coordinates of all atoms) as illustrated in Figure 1.1. This problem is called the *protein structure prediction* problem. A closely related problem is the *protein folding* problem, which is concerned with the prediction of the atomic positions during the actual folding of the protein. A solution to the protein folding problem is therefore also a solution to the protein structure prediction problem and is therefore considered to be a much harder problem. Even though scientists have been trying to solve the protein structure prediction problem since the 1960's, no algorithm is able to predict the structure of proteins in general (in reasonable time). The database of amino acid sequences with known structures (PDB) is growing rapidly. So, algorithms that are based on so-called homology modeling are often able to predict the structure of proteins that have a homolog counterpart in the database. In general, however, we cannot assume that a protein has a homolog counterpart with known structure and so-called *ab initio* or *de novo* algorithms are therefore trying to use more fundamental properties that do not require the specific knowledge of other proteins.

Progress in the field of protein structure prediction will also have a great medical relevance. If we learn how the amino acid sequence is related to the

native protein structure, engineers could design new enzymes working as drugs for various diseases [85]. Protein design is in principle the reverse of protein structure prediction.

In this thesis, several classical algorithms for protein structure prediction and model quality assesments are described together with our latest research.

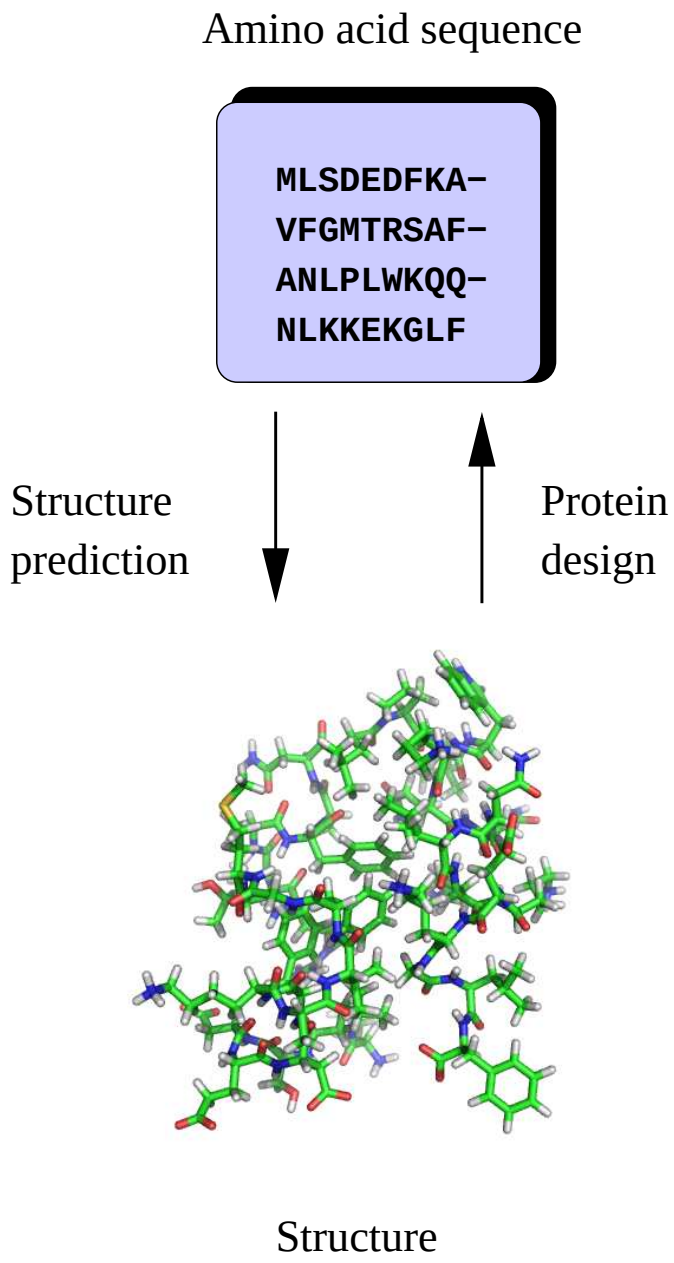


Figure 1.1: Protein structure prediction and protein design.

Chapter 2

Proteins

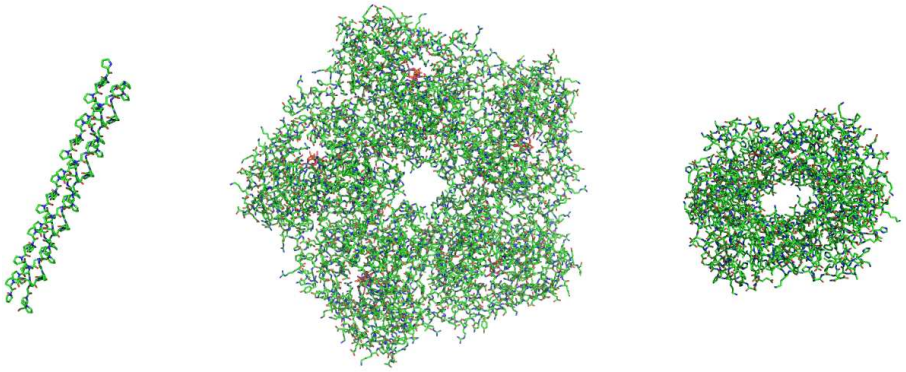
2.1 Categories of Proteins

The diversity of protein functionality is very large and proteins are typically grouped into three categories:

- **Structural proteins.** These are building blocks in skin, hair, nails, muscles etc. An example is the protein *collagen*, Figure 2.1(A), which is bundled in collagen fibers and is the main component in skin, bones and teeth.
- **Enzymatic proteins.** These proteins are catalysts in chemical reactions. An example is the enzyme *glutamine synthetase*, Figure 2.1 (B), which plays an important role in the metabolism.
- **Functional proteins.** These proteins can be thought of as small machines with a very specific task. An example is the protein *hemoglobin*, Figure 2.1 (C), which is responsible for transportation of oxygen in blood.

2.2 Protein Synthesis

A protein is a chain of smaller molecules, called amino acids. Proteins consists of 20 standard amino acids, but there are more of them in nature. Proteins are distinguished by their specific sequence of amino acids. The chain of amino acids is assembled in the living cell - this process is called *protein synthesis*. As illustrated in Figure 2.2, protein synthesis begins in the nucleus with transcription. Here, one part of the *deoxyribonucleic acid* (DNA) strand is copied and encoded into a molecule called messenger *ribonucleic acid* (mRNA). The mRNA is then transported out of the nucleus membrane and translated into a chain of amino acids by the ribosomes in the cytoplasm. The translation of mRNA nucleic acids into amino acids is specified by rules called the *genetic code*. There are four different nucleotides, so mRNA (and DNA) can be represented by strings of an alphabet of 4 letters. Usually A,G,C and T for DNA and A, G, C and U for mRNA. The genetic code is a mapping from every triplet of nucleotides -



(A) (B) (C)

Figure 2.1: (A) Collagen fibers (structural protein). (B) Glutamine synthetase (enzyme). (C) Hemoglobin (functional protein).

giving 64 combinations - into one of the 20 amino acids as illustrated in Figure 2.3. The construction of the actual amino acid chain, also called the polypeptide chain, occurs when neighbouring amino acids react and create peptide bonds (see Figure 2.4).

2.3 Protein Structure

The atoms in the peptide bonds are fixed in a plane called the amid plane. So, the main flexibility of the polypeptide chain, comes from the backbone bonds connected to each C_α atom (see Figure 2.5). A proteins degree of freedom is not only given by the ϕ and ψ angles - most of the side chains also have bonds with several possible dihedral angles. However, modifications in the side chains are local compared to the ϕ and ψ angles which describe the overall path of the backbone chain.

It is generally believed that already during translation the polypeptide chain begins to fold. When the polypeptide chain folds, the amino acids move according to the degree of freedom, such that the Gibbs free energy of the system is minimized. This is described in more detail in section 2.4. In most cases, the chain folds to the *native* structure in the order of milliseconds [12].

2.3.1 Levels of Protein Structure

It is the atomic configuration of the native structure that determines the properties and functionality of a protein. Protein structure is typically described in four levels:

- **Primary structure.** The polypeptide chain is made from amino acids in the ribosomes and the primary structure of a protein describes this

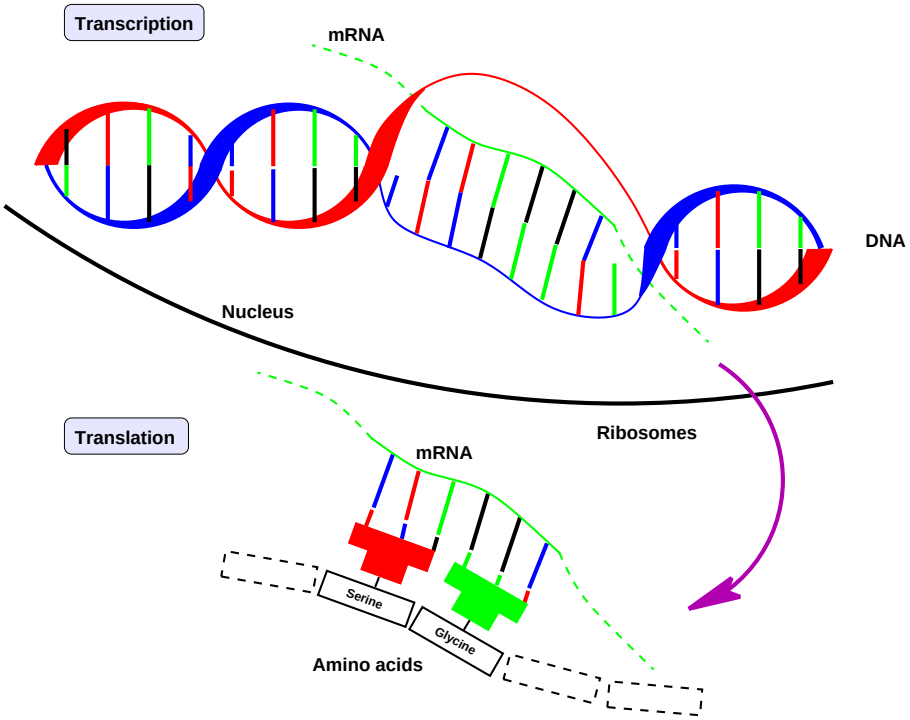


Figure 2.2: Protein synthesis begins with transcription, where a part of DNA is copied to a complementary mRNA molecule. The mRNA molecule is then used in the translation phase where the polypeptide chain is constructed.

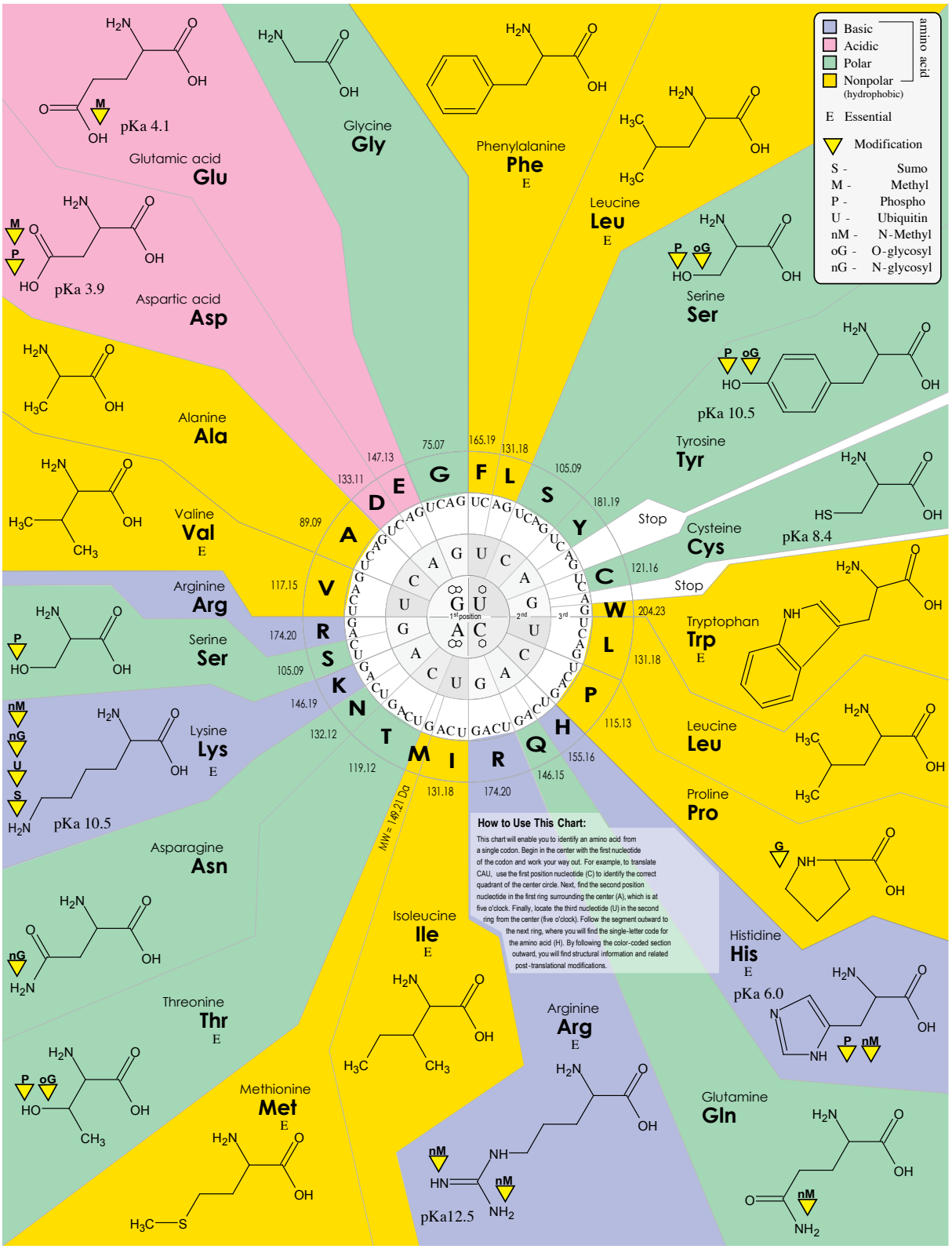


Figure 2.3: Illustration of the genetic code. Image from Wikipedia (http://en.wikipedia.org/wiki/Image:GeneticCode21.svg).

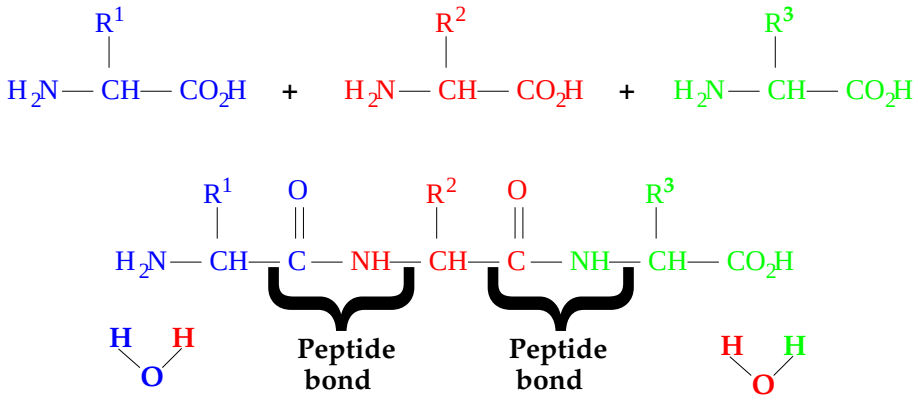


Figure 2.4: Three amino acids are joined by two peptide bonds. In this reaction, two water molecules are released. The R^i-molecules correspond to the side-chains of the amino acids.

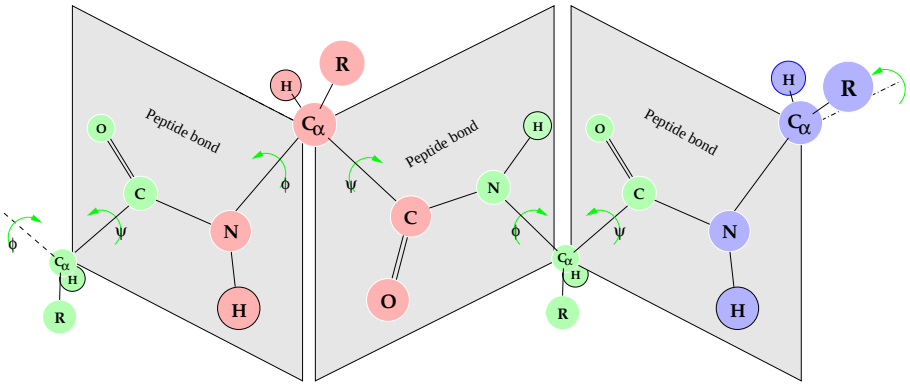


Figure 2.5: Structure of the polypeptide backbone. The backbone atoms in a peptide bond are all fixed in a plane and these planes can *rotate* according to the ϕ and ψ angles. ϕ is the dihedral angle between N and C_α and ψ is the dihedral angle between C and C_α. Each internal amino acid therefore contributes with two degrees of freedom and the end-chain amino acids contribute with one degree of freedom. The additional degree of freedom of the amino acid side-chains depend on the amino acid type.

sequence of amino acids. The primary structure starts at the N-terminal.

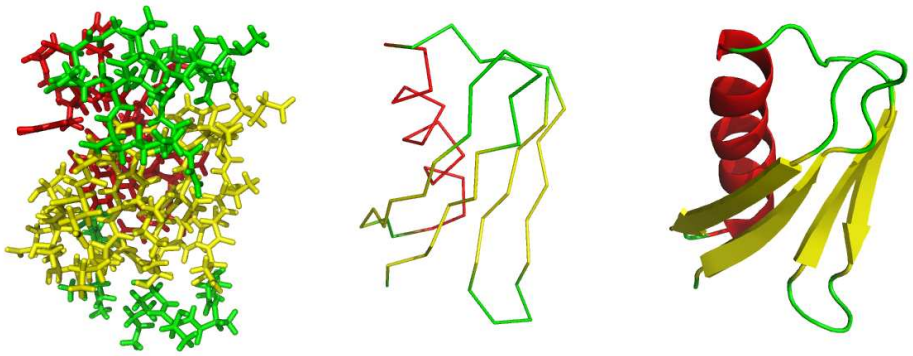
- **Secondary structure.** Some local structure patterns are very often observed in proteins because they lead to stable, low energy, structures. The most frequent of those are the *alpha helices* and the *beta sheets*, but loops and other helices are usually also considered to be secondary structure. The local structure of a backbone that is not a secondary structure element is called a *random coil*. In Figure 2.6 three different visual representations of *protein G* is shown. Protein G contains one alpha helix and one beta sheet with 4 beta strands. It is difficult to see these secondary structures when all protein bonds are printed (A), however in Figure 2.6(B) and Figure 2.6(C) only the backbone atoms are printed and the secondary structure patterns appear more clearly. The arrangement of a specific secondary structure combination is called a *motif* (i.e. sheet, turn, sheet) and a more general description of a secondary structure arrangement is called a *fold*. An example of a typical fold is the *beta-barrel* as illustrated in Figure 2.7.

- **Tertiary structure.** This is the full description of a folded polypeptide chain. In principle the tertiary structure is the 3D-coordinates of the atoms in the native structure of the protein. However, proteins are often not completely stable, so these 3D-coordinates often correspond to the most observed state or the crystallized state of the protein.

- **Quarternary structure.** Some proteins consist of several polypeptide chains which are assembled in a more complex molecule (often called a protein complex). The description of how these folded polypeptide chains are assembled is called the quarternary structure of the protein.

2.4 Thermodynamic Hypothesis

In the late 50' and early 60' Anfinsen et al. [5] studied what happens to the protein ribonuclease when it is first denatured (unfolded) and thereafter renaturated (folded). They observed that different conformations of the unfolded polypeptide chain of ribonuclease always fold to the same native state and they postulated the *thermodynamic hypothesis*. It states that the native state of a protein, in its normal environment, is the structure with lowest Gibbs free energy. This property is fundamental for the understanding of protein folding and is why it is believed that the native state of a protein can be predicted just from the knowledge of its amino acid sequence. Note that the thermodynamic hypothesis has been verified on many different proteins later, however it is observed that some proteins receive help to fold from specialized proteins called chaperones [21]. It is still discussed whether or not the existence of chaperones conflicts with Anfinsens thermodynamic hypothesis.



(A) (B) (C)

Figure 2.6: 3 different visualizations of *protein G*. (A) Sticks, (B) Backbone trace, (C) Cartoon. The sticks visualization shows a *stick* between pairs of bonded atoms. The backbone visualization shows lines between atoms on the backbone. The cartoon visualization clearly shows helices strands and coils in a stylistic figure.

2.5 Protein Folding

If we assume that Anfinsens thermodynamic hypothesis is valid, then proteins contain everything that is needed to fold to the structure with minimum free energy - the native structure. When the chain folds, it must undergo changes mainly in the neighbouring bonds of the C_α-atoms but it is not known in details how these changes occur. The modifications of the chain over time correspond to some path in the energy landscape (Figure 2.8). However, it is not known how many folding pathways exist for a given protein and how these pathways are found by the protein.

2.5.1 Levinthal's Paradox

In 1969 Levinthal [51] argued that proteins could not use a completely random search or exhaustive search since the number of possible conformations of an amino acid chain is astronomical high. This argument is called *the Levinthal paradox* even though it was never believed that proteins actually do fold using completely random search or exhaustive search. Instead, it is very likely that the energy landscape is funnel-like such that the forces acting on an arbitrary unfolded chain, eventually move the atoms in their native positions without having to cross very large energy barriers [19].

2.5.2 Properties of the Energy Landscape

If the thermodynamic hypothesis is perfectly true, then the energy landscape must have the following properties:

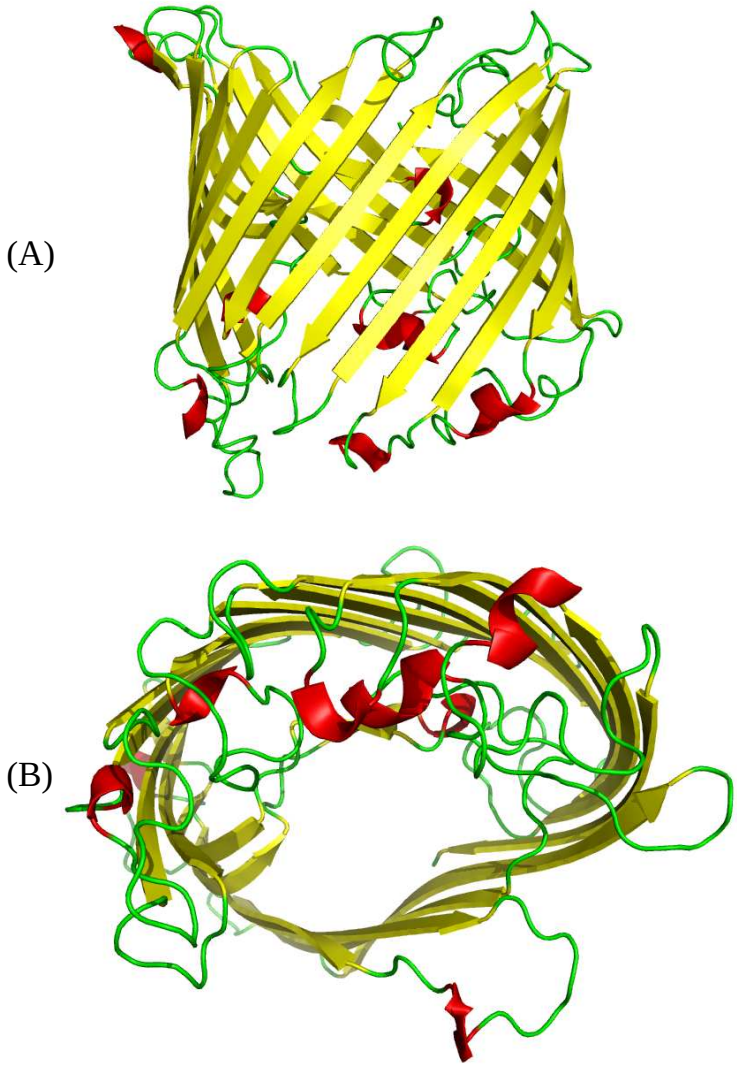


Figure 2.7: The Figure shows one chain of the sucrose-specific porin (PDB: 1A0S). (A) shows the typical beta-barrel fold where beta-strands are arranged antiparrallel. (B) the protein has a hollow center like a barrel.

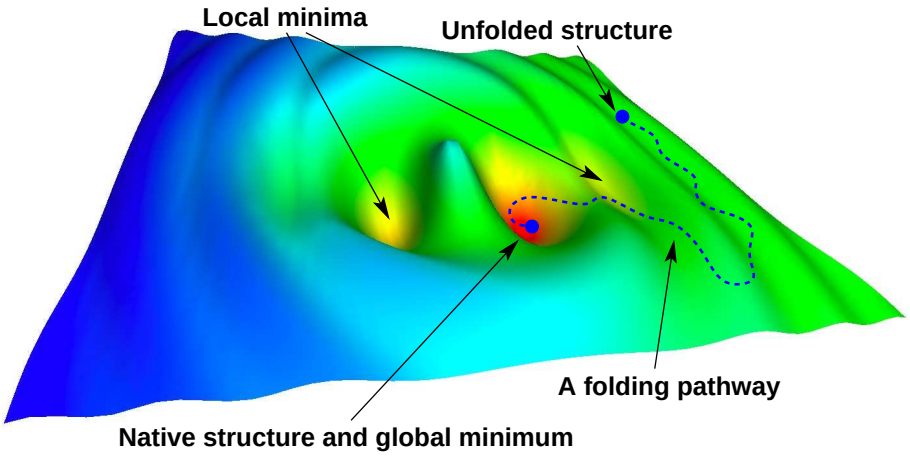


Figure 2.8: Illustration of an energy landscape and a folding pathway. For proteins, the energy landscape corresponds to a multi-dimensional hypersurface that depends on the positions of the atoms. Each point in the energy landscape therefore corresponds to a conformation with an associated energy.

1. **Uniqueness.** There should be only one structure with minimum free energy. Otherwise a polypeptide chain could have several native structures.
2. **Stability.** Small changes in the structure should not give rise to large energy changes. It is known that proteins often fluctuate around the native state. This fluctuation would require much energy if there were large energy barriers around the native structure.
3. **Accessibility.** The path from an unfolded state to the native state should not contain very large barriers. It is assumed that virtually all unfolded states of the polypeptide chain are able to reach the native state. Crossing very large barriers require much energy and it would be difficult, or impossible, for the polypeptide chain to reach the native state.

In Figure 2.9 an illustration of a protein folding pathway is shown. The figure shows 6 snapshots from an unfolded state to the native structure of the protein.

2.6 Chapter Summary

Proteins are complex molecules that exist in all living organisms. They are constructed in the cell in a process called *protein synthesis*. Proteins are chains of smaller molecules called amino acids. There are 20 different amino acids in nature and it is the sequence of amino acids of the protein that eventually determines the shape and functionality of the protein. Typically, in the order of milliseconds, the chain of amino acids folds to the native structure of the protein. It is believed that the native structure has minimum free energy. It is not known in details how proteins move from an unfolded structure to the

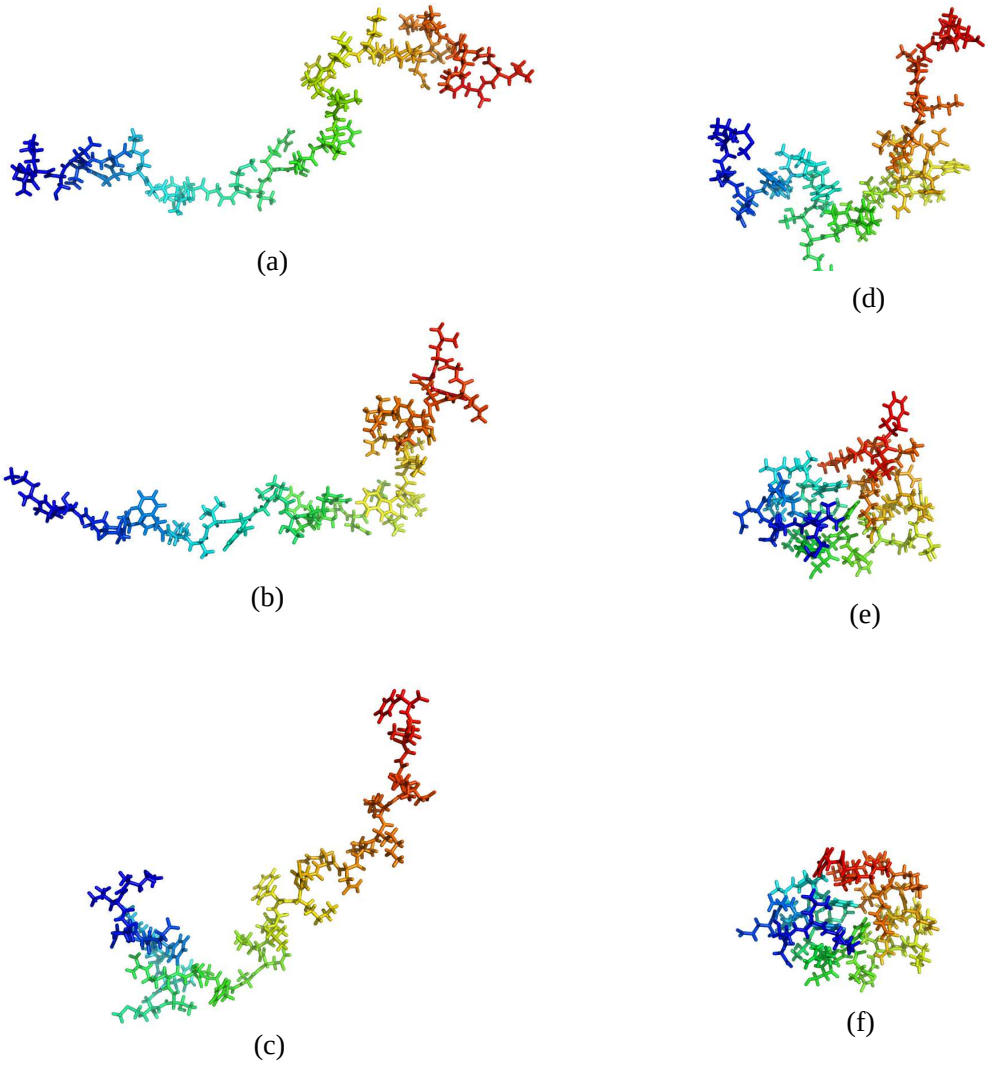


Figure 2.9: What folding of Villin Headpiece might look like in 6 snapshots. From an unfolded state (a) to the folded native structure of Villin Headpiece (f). *Note that these structures are found by simulating an unfolding process from (f) to (a) and are therefore not snapshots of a real protein folding. The structures are therefore just a qualified guess of how the steps of a protein folding could look like. Simulation data is provided by Kresten Lindorff-Larsen.*

native structure, but it is believed that the forces of nature create a funnel-like energy landscape.

Chapter 3

Protein Structure Prediction - An Introduction

In the previous chapter, we described the experiments by Anfinsen and argued that all information needed for folding a protein is contained in the amino acid sequence. In the right environment, the chain of amino acids can therefore be considered as a self-assembling machine. When all information needed is stored in the amino acid sequence, a compelling idea is of course to write an algorithm that takes the amino acid sequence as input and outputs the tertiary structure of the protein. In this chapter, we briefly introduce the various concepts of protein structure prediction and describe the two major problems that we are faced with. This chapter also describes various methods for evaluation of prediction quality. The next chapters 4 and 5 describe in more detail secondary structure prediction and tertiary structure prediction respectively.

3.1 Protein Folding Problem

There are basically two very different approaches for computing the native structure of a protein. One approach is to mimic nature such that the actual folding pathway is computed. The idea is to start with an unfolded chain and apply a physics based energy function (Figure 3.1) such that the atoms move according to Newton's laws of motion. If this can be done with enough accuracy and speed, the simulation would eventually reach a structure similar to the native structure of the protein. This problem is called *the protein folding problem* and is often studied using molecular dynamics. A solution to the protein folding problem would therefore give both the atomic pathways of folding and the native structure of the protein. The main difficulty with the naïve molecular dynamics approach is that the forces acting on the atoms are not constant, they depend on the positions of all atoms. All simulations using this approach are therefore approximations of the real pathway. Experiments show that it requires extremely small time steps to give a realistic simulation and small timesteps require many computations of the forces acting on the atoms. The longest protein that has been correctly folded using molecular dynamics is the Villin Headpiece. It has 36 residues, and using the *folding@home* massively distributed program,

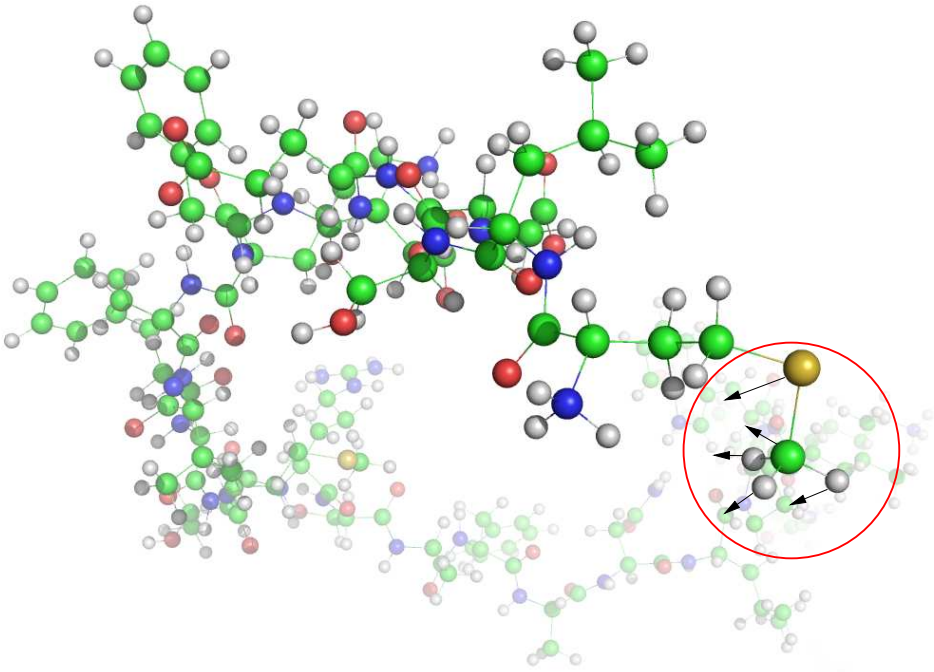


Figure 3.1: The forces acting on the atoms are computed and the positions of the atoms are moved accordingly.

the Pande group was able to simulate 500 μs of folding using 200.000 CPU's[32]. With exceptions of the smallest peptides, this approach is therefore currently not feasible.

3.2 Protein Structure Prediction

Fortunately, for most applications the knowledge of the native structure is more important than the folding pathway. Another main approach is therefore to disregard the actual folding pathway and predict the native structure from amino acid sequence using another method than nature does. This problem is called *the protein structure prediction problem*. Both of these problems are still unsolved. The main reasons for this are the following two obstacles.

3.3 Major Obstacle 1: Energy Function

We currently believe that in nature, the motion of the atoms are described by *quantum mechanical* (QM) rules. The most accurate and obvious choice of energy function would therefore be a QM-based energy function. However, even for small molecules (like a single amino acid) the QM-based energy is very difficult and time demanding to compute. Since proteins are very large molecules, the use of purely QM-based energy functions are currently not feasible. Another approach is therefore to use energy functions that are easier to compute and to

some extend approximate the real QM energy of the protein. Many of the popular approximate energy functions used today are weighted functions of several energy terms, i.e. bond energies, electrostatic forces, van der Waals forces etc. Each of these functions depends on pairs of bonded or non-bonded atoms. These functions are of course fast to compute, but it is a crude assumption that the energy of a protein can be described as a sum of functions that only depend on pairs of atom positions.

The side chains of amino acids are either *hydrophobic* or *hydrophilic* because of their polarity. Consequently, since many proteins are water soluble the main forces in protein folding are concerned with the hydrophobic packing of the protein core. Water soluble proteins therefore tend to have the hydrophobic (nonpolar) amino acids buried in the core of the protein where they are isolated from water. Even though the hydrophobic/hydrophilic forces are consequences of the natural QM energy function, they are often handled by rewarding hydrophobic core packing explicitly [48].

3.4 Major Obstacle 2: The Conformational Space

The conformational space of a protein is huge. In Section 2.3 we showed that each amino acid in the chain gives rise to two degrees of freedom (not counting the degrees of freedom of the side-chain). So, even crude discretizations of the degrees of freedom result in an astronomical large conformational space. Finding the structure with minimum free energy can therefore not be done using complete enumeration. However, in [67] (described in more details in Section 7.5.2), we show that it is possible to exploit the structure of our energy function to efficiently bound large regions of the conformational space. We therefore consider these two obstacles to be intertwined.

3.5 Other Structure Prediction Problems

It is easier to predict the *secondary structure* of a protein compared to the *tertiary structure*. The reason for this is that secondary structure is mainly considered to be a local property of the chain. As described in the next chapter, neural networks using a local window of amino acids are successful in predicting secondary structures. Another category of structure prediction is the side-chain positioning problem [22] (Figure 3.2). This problem is also easier than the tertiary structure prediction. The main reason is that only a few configurations of each side-chain are energy favorable. These configurations are called rotamers and algorithms like SCWRL[14] are able to assign rotamers to the side-chains of a backbone with high accuracy in reasonable time.

3.6 Methods for Evaluation of Predictions

To date, no algorithm is able to *exactly* compute the secondary, tertiary or quarternary structure from the primary structure. Nevertheless, algorithms can predict these structures with various degrees of accuracy. In the literature,

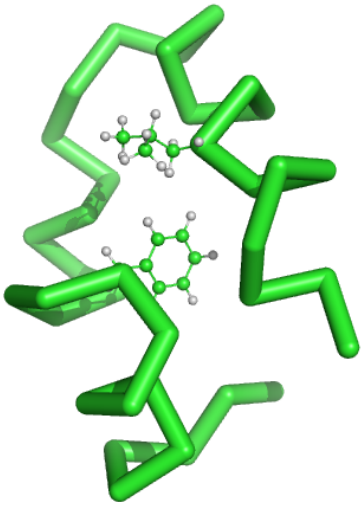


Figure 3.2: In the side-chain positioning problem, the whole backbone is fixed and the problem is to assign the correct orientation (rotamer) of the side-chains. Here two of the side-chains with correct rotamers are shown.

different measures for evaluation of prediction quality have been proposed. Here we describe two measures for secondary structure prediction evaluation (Q_3, CC) and three measures for tertiary structure prediction (RMSD, GDT and AC). In all cases, these measures quantify the similarity between two structures in a single number.

3.6.1 Q_3

A simple and widely used measure for secondary structure prediction quality is the Q_3 score [9]. The Q_3 score of a secondary structure classification is simply the percentage of correctly predicted assignments of secondary structure. Figure 3.3 shows an example of a secondary structure prediction being compared with the exact secondary structure derived from the native structure. The correct assignments are highlighted. While the Q_3 score is easy to compute and interpret, it is not always the best indicator of the prediction quality. Most proteins contain more coil structure than helical and sheet structure and an overprediction of coil is therefore not penalized properly. A prediction algorithm that only predicts coil would, on average, achieve a higher Q_3 score than a prediction algorithm that only predicts helices which is not reasonable.

3.6.2 CC

A perhaps better method for evaluation of secondary structure prediction is the correlation coefficient (CC) [56]. The CC is a number between -1 and 1 and is defined as follows.

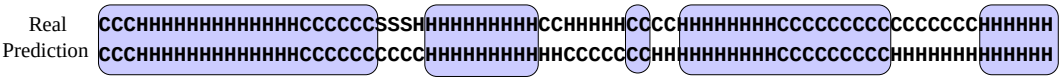


Figure 3.3: An example of a secondary structure prediction together with the exact secondary structure derived from the native state of the protein. H corresponds to *helix*, S corresponds to *sheet* and C corresponds to *coil*. The Q_3 score in this example is $56/76 \simeq 74\%$. The shown example is the secondary structure prediction of Calbindin (4ICB) using PSIPRED [57]. This prediction is used in our algorithms [67, 66, 24] described in Chapter 7.

$$CC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FN)(TP + FN)(TN + FP)(TN + FN)}}$$

where TP is the number of true positives, TN is the number true negatives, FP is the number of false positives and FN is the number of false negatives. If we want to measure the CC of helical prediction, TP would correspond to the number of correctly predicted helices, TN the number of correctly predicted non-helices, FP the number of wrongly predicted helices and FN the number of wrongly predicted non-helices. A CC of -1 means that all predictions are wrong. A CC of zero means that the prediction is 'random' and a CC of 1 is a perfect prediction.

3.6.3 RMSD

It is often important to measure the similarity between two protein backbones. In some search algorithms we only want to consider protein backbones that are different to some extent [63] or perhaps we want to compare a predicted structure with the native structure. The *Root Mean Square Deviation* (RMSD) measure is often used for backbone prediction quality. In literature, two versions of RMSD are generally used:

- Coordinate RMSD is

$$CRMSD = \sqrt{\frac{\sum_{i=1}^{n} |\bar{a_i} - \bar{b_i}|^2}{n}}$$

  where a and b are two vectors of coordinates (usually C_α-coordinates) that should be compared. CRMSD can be computed for all positions of a and b in space, however CRMSD is usually computed for the optimal superposition between a and b. The task of finding an optimal superposition of two point sets can be tackled by various techniques. One technique used in many implementations is the quaternion method described by S. Kearsley [39]. It runs in $O(n)$ time where n is the length of the vectors being compared.

- Distance RMSD is

$$DRMSD = \sqrt{\frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|\bar{a_i} - \bar{a_j}| - |b_i - b_j|)^2}{(n(n-1))/2}}$$

  This definition of DRMSD does not depend on the relative positions of a and b, and it is therefore not necessary to compute the optimal superposition.

3.6.4 GDT

It has long been known by CASP organizers and CASP assessors that RMSD is not always well-suited for evaluation of protein structure predictions, especially long and difficult targets. The reason is that RMSD is a global measure that does not reward partially correct models. To illustrate the problem with RMSD, consider a structure where 70% of the first residues can be aligned with the target and the last 30% is positioned completely different from the target (Figure 3.4). Often such a prediction is considered to be a good prediction (if no close homologoues are known), however, RMSD does not reflect this very well. The Global Distance Test (GDT) [97] measure was proposed in 1999, and it avoids this problem. Basically, GDT computes the percentage of residues having C_α-atoms that are within a certain distance cutoff from the target in an optimal superposition. With an appropriate distance cutoff, GDT would therefore be 70 in the example illustrated in Figure 3.4. By varying the cutoff-value, a very descriptive plot of structural similarity can be generated as in the example shown in Figure 3.5. It is often not trivial to decide what the distance cutoff should be when GDT is used as a score function. The GDT score is therefore often reported using an average of different cutoff distances; typically 1, 2, 4 and 8 Å.

3.6.5 AC

None of the measures described above consider the positions of side chains. For most applications, it is more important to correctly predict the overall backbone path i.e. C_α-trace than correctly predict the positions of side chains. In our paper *Reconstructing protein structure from solvent exposure using tabu search* [63] we introduced a very simple measure for evaluating the implicit directions of side chains. Given only the C_α-trace, we wanted a simple and fast measure to evaluate if side-chains eventually would be pointing out of the protein (surrounded by water) or pointing towards the interior of the protein. We came up with the following *angle correlation* (AC) that has the following definition

$$AC = \frac{\sum_{i=1}^{n}\theta(\bar{a_i}, \bar{b_i}) - \theta(\bar{c_i}, \bar{d_i})}{n}$$

where $\bar{a_i}$ is the vector pointing from the i'th C_α-atom to the geometric center and $\bar{b_i}$ is the vector pointing in the direction of the i'th C_α-atoms side chain (as defined in Figure 3.6). Vectors c_i and d_i are the corresponding vectors

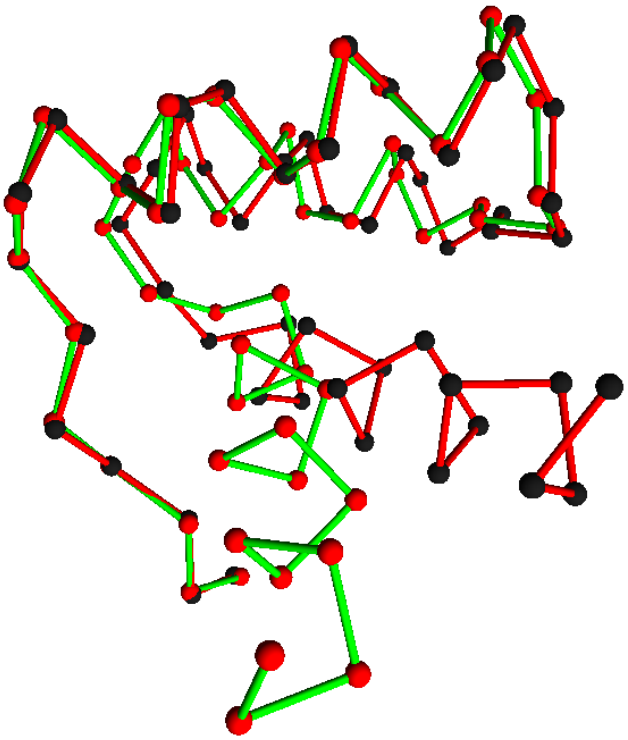


Figure 3.4: Two C_α-traces have almost equal positions of C_α-atoms when the first 70% of the residues are compared. This gives a GDT$\simeq$70 even for small cutoff distances. The RMSD of the best superposition of all C_α-atoms is 3.1. If RMSD is measured for the 70 % of the first residues only, then RMSD would be 0.3.

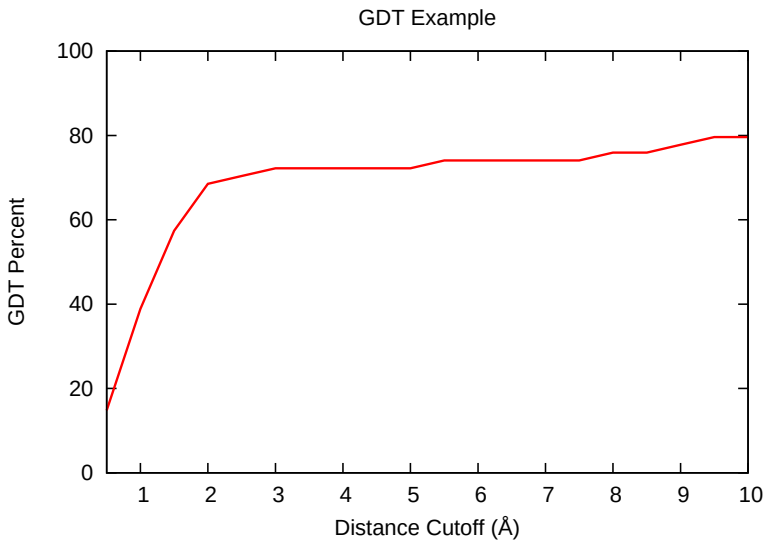


Figure 3.5: GDT is computed for various distance cutoffs. The plot shows that for cutoff distances (>2 Å) the similarity between the structures is almost stable on 70% in this example.

in the native structure. θ measures the angle between the two vectors. Zero AC is perfect correlation, 90° is random correlation and 180° is perfect 'anti'-correlation. There are many examples where a structure can have a good RMSD and a bad AC (or the opposite). This can be problematic when side-chains are positioned on structures that we only selected because of good RMSD or good GDT. In these cases, side-chain positioning might be impossible or perhaps give a wrong full tertiary structure. Figure 3.7 shows a typical RMSD vs. AC plot from [63]. In Figure 3.8 superpositions of two structures with the native structure is shown. Both structures have good RMSD but structure (a) has bad AC and structure (b) has good AC.

3.7 Critical Assessment of Techniques for Protein Structure Prediction (CASP)

Critical Assessment of Techniques for Protein Structure Prediction (CASP) [60] is an experiment that determines the current state of art for protein structure prediction algorithms. Every two years the CASP organizers collect unpublished protein structures from crystallographers. The main idea is therefore to blind test current algorithms for protein structure prediction on these *secret* protein structures. The predictors (participants) of CASP therefore only know the amino acid sequences of these proteins and are asked to predict the native structure using their algorithm. When the predictions have been submitted to the CASP organizers, the secret protein structures are revealed and the different algorithms are evaluated.

There are different evaluation categories at CASP. The one just described

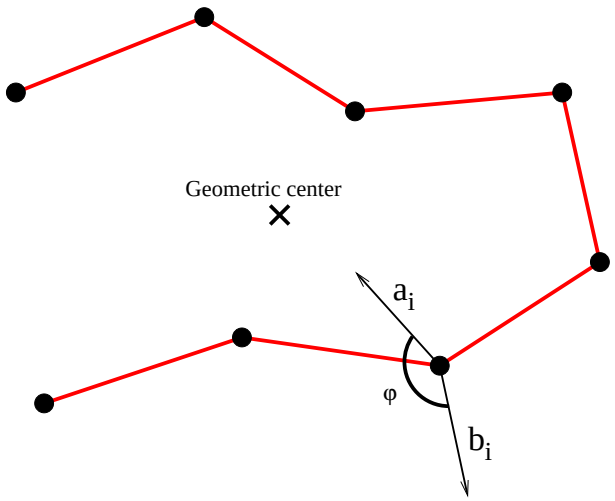


Figure 3.6: An illustration of the vectors involved in the computation of AC.

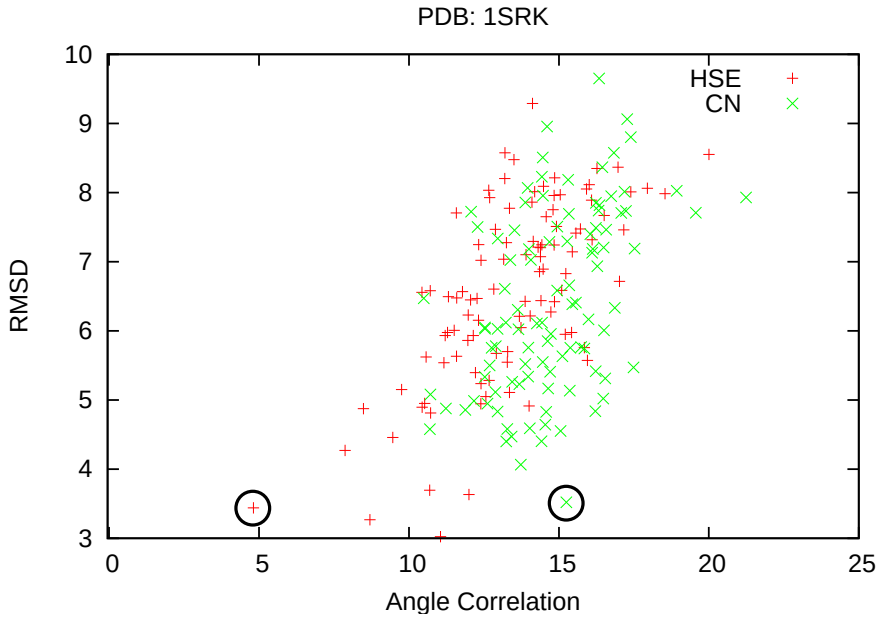
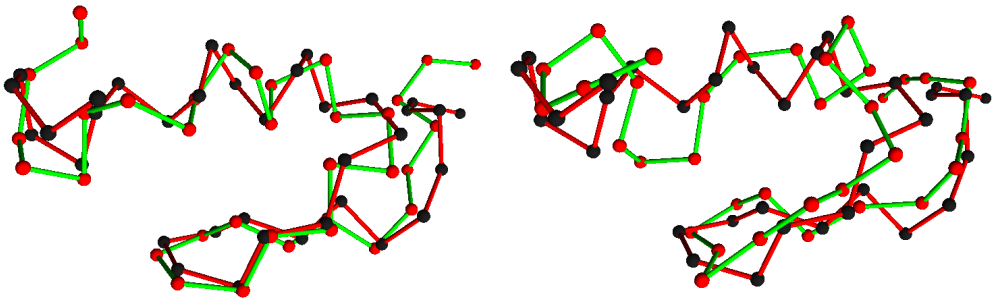


Figure 3.7: An RMSD vs AC plot from [63]. In this experiment, 100 low energy structures are found using two different energy functions (HSE-based and CN-based). The HSE-low-energy structures tend to have a slightly better RMSD and a significant better AC than the CN-optimized structures. The structures marked with circles are shown in Figure 3.8.



(a) (b)

Figure 3.8: The red structure is the C_α-trace of 1SRK and the green structures are the C_α-traces of two of our predictions. The left green prediction is the HSE-optimized structure and the right green prediction is the CN-optimized structure. The optimized structures are marked with circles in Figure 3.8. Both predictions have similar RMSD but different AC.

is the so-called tertiary structure prediction category, which is also considered to be the most important since protein structure determines protein function. Another category is the *model quality assessment* (MQA) category. Here a set of alternative structures (models) for the same amino acid sequence is given. The task of the MQA predictors is therefore to estimate the quality of these models without knowing the native structure of the protein. A good MQA algorithm would therefore be able to rank the alternative models correctly. This is important because most algorithms for protein structure prediction do not only predict one structure but a whole set of structures. MQA is described in more details in Chapter 6.

3.8 Our Research

In our research projects, we do not consider the actual protein folding problem. To attack obstacle 1, our energy functions are all based on knowledge based measures found by algorithms that are trained on a large set of proteins. When databases of proteins with known structures grow we also expect our energy function to be more accurate. Our energy functions are described in more detail in Chapter 7. To overcome obstacle 2, we apply a advanced search heuristics (Section 7.5.1) and in Section 7.5.2 we describe our approach for implicit sampling of the whole conformational space.

We use most of the methods for evaluations of predictions described here.

In [63, 62] the predicted structures are very small (up to 35 residues) and the global measure, RMSD, is appropriate for evaluation of prediction quality. We also show that the so-called HSE optimized structures have a better angle correlation using the AC measure. In [67, 66, 24, 65] we obtain the secondary structure prediction from PSIPRED [57]. Even though we argued that the CC in many cases is more appropriate for measuring secondary structure prediction quality, we use the Q_3 instead. The reason for this is, that we want to show that the secondary structure predictions are comparable in quality with other PSIPRED predictions which are usually evaluated using Q_3. Another reason is that people are also more confident with the Q_3 score because it is easier to interpret. In our MQA approach [64], we use the GDT measure to evaluate our correlation between predicted quality and real quality (GDT). We currently participate in CASP8 with our MQA approach which is described in more detail in Section 6.3.

3.9 Chapter Summary

In the right environment, the amino acid chain is a self-assembling machine. All information needed to compute the tertiary structure is stored in the amino acid sequence. The problem of computing the folding pathway of the amino acid chain from an unfolded structure to the native structure is called the protein folding problem. An easier problem is to disregard the folding pathway and *just* compute the native structure of the protein given the amino acid sequence. Even though both problems are extremely important they are still unsolved. There are two main reasons for this. One problem is that the natural energy function is very difficult or even impossible to compute and all known approximations of the natural energy function do not have the required properties. Another problem is that the conformational space is very big and difficult to search. Even though these problems have not been solved, there is a vast number algorithms for computing folding pathways and predicting protein structure. However, either they never give reasonable results or they can only predict the tertiary structure of a small subset of proteins. For evaluation of algorithms for protein structure prediction, many measures have been proposed. In this chapter, some of the most popular measures are described together with their advantages and disadvantages. Every second year the state-of-the-art algorithms are assessed at CASP. Here, the algorithms are blind-tested on a number of protein structures and their tertiary structure prediction are evaluated when the native structures of the proteins are revealed.

Chapter 4

Secondary Structure Prediction

Given an amino acid sequence, secondary structure prediction is the task of mapping each amino acid into one of the secondary structure categories. These categories are typically helix, strand and coil. The first secondary structure prediction algorithms, developed in the 1960s and 1970s, were based on the properties of the single amino acids [28, 52, 81], even though there are some correlation between amino acid type and secondary structure, the accuracy of these algorithms is poor. Later, a significant improvement in accuracy came when the algorithms considered segments of contiguous amino acids for prediction of secondary structure. Among the best performing algorithms in the 1980's [78] and 1990's [82] were algorithms based on neural networks. Today, one of the best performing algorithms (PSIPRED) is based on feed-forward neural networks combined with evolutionary information [57]. In the following text it is briefly described how the feed-forward neural network can be used for prediction of secondary structure.

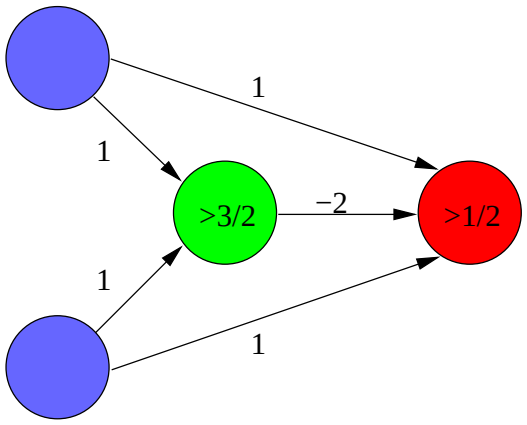
4.1 Neural Networks

Artificial neural networks are usually very rough models of natural neural networks that exist in most animals. These biological neural networks (brains) are very complex and are not considered in this text. However, the terminology used to describe artificial neural networks use some of the words that describe biological neural networks (like neurons and synapses). In the following text, an artificial neural network is therefore just referred to as a *neural network*.

The main motivation for using a neural network for solving biological problems compared to other methods is their ability to handle inconsistent and noisy data.

4.1.1 The Neuron and the Synapses

The simplest unit in a neural network is the neuron. The neurons can be connected to each other, and these connections are called *synapses*. A synapse has a weight called the *synaptic strength*. The synaptic strengths are usually determined by training (Section 4.2). The neuron has a transfer function that



XOR	0	1
0	0	1
1	1	0

Figure 4.1: This neural network can calculate the XOR function. The two left-most neurons are *input* neurons and the right-most neuron is the *output* neuron. In this example, the values of the input neurons are binary. The value of the other neurons are calculated according to Equation 4.1. The transfer function of the neurons are step functions returning either 0 or 1 depending on the threshold of the neurons (shown on the non-input neurons). The synaptic strengths are shown on the edges and the XOR function is shown in the table. The neurons in this example do not have biases.

determines the value of the neuron based on the values of the other incoming neurons together with the synaptic strengths. In standard implementations, the value of neuron i is:

$$y_i = f\left(\sum_j v_{ji} y_j + v_i\right) \tag{4.1}$$

where f is the transfer function, v_{ji} is the synaptic strength from neuron j to neuron i, v_i is the neurons bias and y_j is the value of neuron j. A small example of a neural network is shown in Figure 4.1.

4.1.2 Transfer Functions

The value of the neuron depends on the transfer function as shown in Equation 4.1. In the example in Figure 4.1, the transfer function is a step function, but more often continuous and differentiable transfer functions are used. One of the most applied transfer functions is the sigmoidal function

$$y(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

In the limits ($\pm\infty$), the sigmodial function behaves like a step function and near $x = 0$ the sigmoidal function is almost linear.

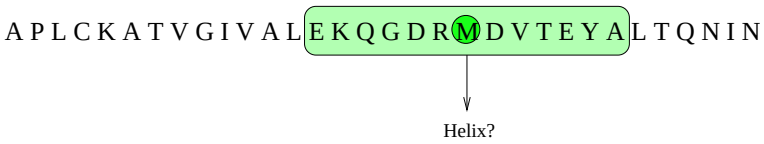


Figure 4.2: A window slides over the amino acid sequence and the neural network predicts the secondary structure of the central amino acid.

4.1.3 Feed-Forward Architecture for Secondary Structure Prediction

This neural network architecture has three layers. An input layer, a hidden layer and an output layer. The input to the neural network is a segment of amino acids called a *window* and the output is the classification of the central amino acid in the window (Figure 4.2). The encoding of an amino acid is *orthogonal*, meaning that there is a neuron for each of the 20 amino acids. Windows at the beginning or the end of an amino acid sequence contain slots with no amino acid, so an additional neuron is used to represent non-existent amino acids. The neurons of the input layer are therefore clustered in groups of 21 neurons as illustrated in Figure 4.3. The size of the window and the number of neurons in the hidden layer are variables and good values must be determined experimentally. The output layer typically has two neurons, one corresponds to the classification of a secondary structure class (i.e. helix) and the other corresponds to other classes (i.e. strand and coil). Similar neural networks for other secondary structure classes can be made. The prediction of an amino acid window can be determined by the *winner takes it all* strategy, meaning that the output neuron with highest value determines the classification.

4.2 Training the Network

When *training the network*, we want to adjust the parameters of the network such that it has optimal performance. In most cases the architecture is fixed and the weights of the synapses are adjusted. When a neural network is used for classifying input data (i.e. a window of amino acids) into secondary structure, it is natural to compare the output of the neural network with the correct classification. For this purpose, *training data* is used to compute an error depending of the network output and the correct output. When training the network, the synaptic weights are therefore adjusted such that this error becomes minimal. The naïve approach to this task is to try all combinations of synaptic weights and choose the set of weights that gives the minimum error. However, in theory the synaptic weights are continuous and even crude discretizations give a huge number of combinations. It is therefore necessary to use another strategy for determining the weights of the synapses.

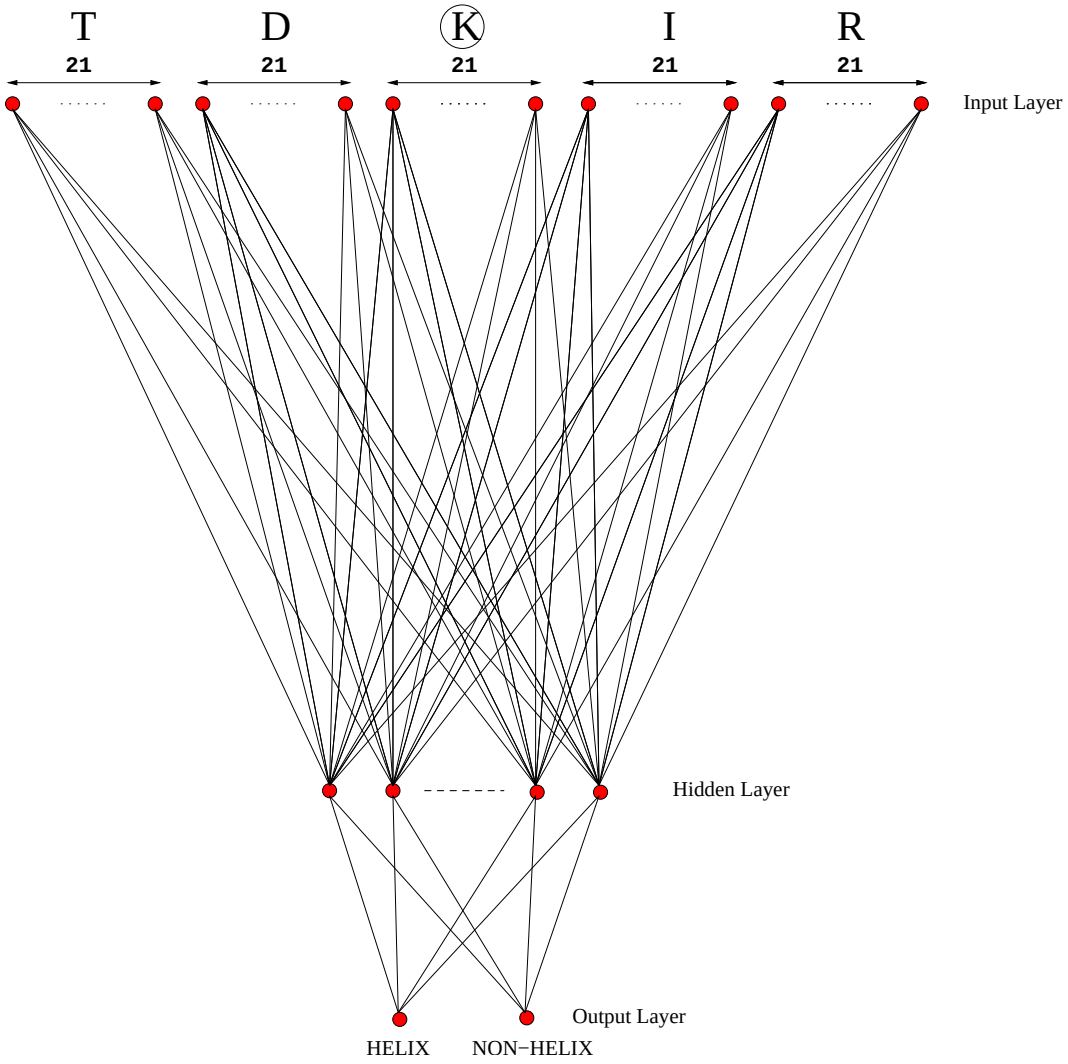


Figure 4.3: Three-layered neural network.

4.2.1 Backpropagation

Backpropagation is an algorithm for adjusting the synaptic weights such that the error becomes small. Backpropagation does not guarantee to find the optimal set of weights, but nevertheless, it is the most applied algorithm for training feed-forward networks. Basically, the backpropagation algorithm computes the gradient of the error function efficiently. The gradient is a vector, that points in the direction where the function grows most. The opposite direction of the gradient therefore indicates where the function declines the most. By calculating the gradient of the error function and adjusting the synaptic weights accordingly it is possible to minimize the error function. Refer to [80] for more details about the backpropagation algorithm.

4.3 Our Research

In most of our research we make use of secondary structure predictions. These predictions are often reliable and contain much information about the native structure of the protein. We do currently not make use of our own neural network for secondary structure prediction (a standard feed-forward network with backpropagation training). Instead, we make use of online webservers because they are more accurate.

In our papers [67, 66, 24], we use predictions of secondary structure to reduce the conformational space. The secondary structure elements are used as rigid segments and it is therefore important that the secondary predictions are as good as possible. We therefore use the PSIPRED webserver for this task, which is based on the feed-forward network and is generally believed to give the best performance. In [67, 66, 24] we also use contact number and half-sphere-exposure measures from neural network predictions. In [64] the contact number probability distributions used by our model quality assessment algorithm are found using feed-forward neural networks. Section 7.5.2 describes in more detail our approach of fixing secondary structure segments and using contact number predictions. Chapter 6 describes in more detail how contact number probabilities are applied for model quality assessment.

4.4 Chapter Summary

Secondary structure prediction is a problem that has received much attention in the literature. Therefore, many different algorithms have been proposed for this problem. In the 1980's neural networks showed great promise for solving this problem. Today, one of the best algorithms (PSIPRED) is based on feed-forward neural networks together with evolutionary information. Secondary structure prediction has many applications and many of the tertiary structure prediction algorithms rely heavily on good secondary structure predictions.

Chapter 5

Tertiary Structure Prediction

Tertiary structure prediction is the task of predicting the native structure of a protein given the amino acid sequence. A vast number of algorithms that attack this problem are described in the literature. In this chapter, some of the fundamental approaches are briefly described (molecular dynamics and homology modeling).

Our approaches for protein structure prediction are based combinatorial optimization. We therefore devote Chapter 7 to prediction algorithms from the literature based on combinatorial optimization and describe our research in this context.

5.1 Molecular Dynamics

The typical way of doing molecular dynamics on proteins is to consider the atoms as spheres with a given radius and mass. The energy function is often a sum of different terms that represent both bonded and non-bonded interactions. The motion of the atoms is then assumed to follow Newton's laws of motion

$$\mathbf{F}_i = m_i \mathbf{a}_i = m_i \frac{\partial^2 \mathbf{r}_i}{\partial t^2} \tag{5.1}$$

where $\mathbf{F}_i$ is the force vector acting on atom i, m_i is the mass of atom i and a_i is the acceleration vector of atom i. The force acting on atom i can also be derived from the potential energy U, such that

$$\mathbf{F}_i = -\frac{\partial}{\partial r_i U} \tag{5.2}$$

When combining these two equations, we get the differential equations that describe the motion of atom i when we know the energy function U

$$m_i \frac{\partial^2 \mathbf{r}_i}{\partial t^2} = -\frac{\partial}{\partial r_i U} \tag{5.3}$$

where $\mathbf{r}_i$ are the coordinates of the i'th atom and t is the time. If Equation 5.3 could be solved analytically for all atoms, we would end up with a function, $\mathbf{r}_i(t)$ for each atom, that describes the exact coordinates of the atoms as a

function of the time given the energy function U. In such case, protein structure prediction would be easy, because the coordinates of the native structure would be $\lim_{t\to\infty} \mathbf{r}_i(t)$. However, proteins are complex systems with thousands of atoms that interact. The energy function therefore depends on the position of all atoms, and no analytical solution are known to exist. Therefore, Equation 5.3 must to be solved numerically. There are many different approaches to how this can be done, however they are all approximations.

5.1.1 The Verlet Algorithm

One of the most straightforward numerical integration algorithms is the Verlet algorithm which is described here. For other more precise algorithms, refer to [76]. The Verlet algorithm, and many other numerical integration algorithms use the Taylor expansions of the energy function:

$$\mathbf{r}_i(t+\delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\delta t + \frac{1}{2}\mathbf{a}_i(t)\delta t^2 \tag{5.4}$$

$$\mathbf{r}_i(t-\delta t) = \mathbf{r}_i(t) - \mathbf{v}_i(t)\delta t + \frac{1}{2}\mathbf{a}_i(t)\delta t^2 \tag{5.5}$$

The sum of these equations is

$$\mathbf{r}_i(t+\delta t) + \mathbf{r}_i(t-\delta t) = 2\mathbf{r}_i(t) + \mathbf{a}_i(t)\delta t^2 \tag{5.6}$$

The position $\mathbf{r}_i$ of atom i a small timestep δt from the current time t is therefore

$$\mathbf{r}_i(t+\delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-\delta t) + \mathbf{a}_i(t)\delta t^2 \tag{5.7}$$

which is a function of the current positions and the previous positions of the atoms.

A deterministic molecular dynamics simulation could therefore start with some unfolded amino acid chain and iteratively update the positions of the atoms using Equation 5.7. One of the problems with this approach (as described in Section 3.1) is that a proper simulation needs extremely small values of δt, typically in the order of femto seconds 10^{-15}. Even though some proteins fold very fast (in the order of micro seconds), molecular dynamics is still not a feasible approach for protein structure prediction. However, molecular dynamics can be a useful tool for learning more about the mechanics of protein folding.

5.2 Homology Modeling

Many proteins have a high degree of structural similarity among different species. These proteins often have important functionalities in the living cell and are therefore needed in many life forms. When protein sequences from different species have a high degree of similarity, they are said to be *conserved* or *homologous*. In the paper by Chotia and Lesk [16] it is shown experimentally that homologous proteins with a high sequence identity are generally more similar in structure than proteins with less sequence identity. The main technique in homology modeling is therefore to find sequences with high sequence identity

to the target sequence in the database of proteins with known structure (i.e. PDB). One or more of these structures are chosen as *templates* and are used to predict the structure of the target.

Many protein sequences have a homologue counterpart in PDB, and for these proteins, homology modeling can be a successful prediction algorithm. Typical steps in homology modeling are, 1) template recognition, 2) target/template alignment, 3) model building and, 4) model quality assessment [55]. In the following sections, procedures 1 to 3 are described briefly. Since we have made some contributions in the field of model quality assessment [64], we devote the next chapter to this topic.

5.2.1 Template Recognition

When searching a database for templates, different algorithms are typically used such that:

- Close homologues are identified with fast and simple algorithms such as FASTA [70] and BLAST [1].
- Remote homologues are identified with more sophisticated algorithms such as SAM-T06 [37] and PSI-BLAST [2].
- No homologues could be found. This might either be because the algorithms cannot detect the homologues or because no homologues exist in the database.

BLAST and SAM-T06 are among the most popular algorithms for finding homologues and are briefly described here. We also use the SAM-T06 server for finding templates in our model quality assessment algorithm.

BLAST

Typically, one needs to query a sequence against a large database with millions of amino acids and detect the sequences with highest alignment score. One way to do this is to run the well-known Smith-Waterman algorithm [88] on the query sequence and all database sequences and return the highest scoring sequence(s). However, because of the size of the databases and the time complexity of the Smith-Waterman algorithm, this approach is often infeasible.

Basic Local Alignment Search Tool (BLAST) was developed by Altschul et al. and published in 1990. Like the Smith-Waterman algorithm, it computes a sequence alignment between two strings (i.e. amino acid sequences or nucleotide sequences) and assigns an alignment score. While the Smith-Waterman algorithm is exact (it always computes the best local alignment) BLAST is a heuristic algorithm and typically runs several orders of magnitude faster than the Smith-Waterman algorithm.

The main increase in speed comes from the fast pre-filtering of sequences. BLAST first checks if the query sequence contains a subsequence (typically three amino acids) that scores at least T when aligned with a subsequence in

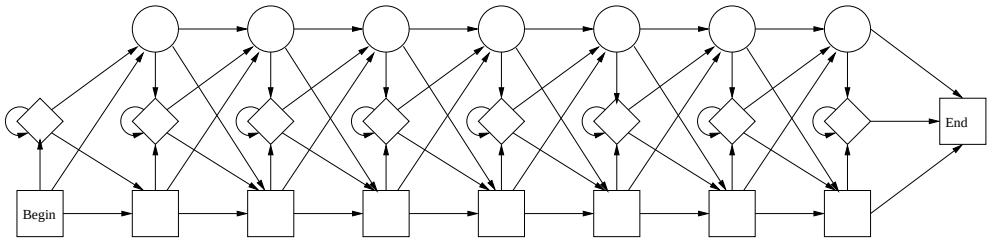


Figure 5.1: A typical HMM topology for biological sequence analysis.

the database sequence. If that is not the case, the database sequence is expected to be insignificant and is discarded. The threshold parameter T therefore determines the sensitivity of BLAST. In the next step, the sequence of three amino acids is extended in both directions to improve the alignment score even further. Again, if the extended alignment score is below some threshold, the database sequence is discarded. Finally, if the database sequence is not pre-filtered, a local alignment is being computed. Using this approach, many of the database sequences that eventually would give a low alignment score are pre-filtered which is much faster than computing the whole local alignment.

HMM for Database Search (SAM-T06)

Sequence Alignment and Modeling system (SAM) [37] is a collection of programs mainly for creating and using hidden Markov models (HMMs). The construction of HMMs and homology detection is automated by the online server called SAM-T06 which is currently the best performing SAM server. The SAM-T06 server can be used for various kinds of local structure prediction and tertiary structure prediction. Our algorithm for extracting distance constraints from alignments (described in Section 6.3) uses templates and alignments found by SAM-T06, SAM-T2K and SAM-T04. The SAM-servers have a lot of features, however, the central part of SAM is the construction and use of HMMs. Here, we therefore only describe how HMMs can be used for template detection.

An HMM is a statistical objects that have been used in many applications, especially speech recognition. In 1994 Krogh et al. [43] described how HMMs can be applied for various tasks in protein modeling. An HMM can be illustrated as a graph, or more specifically, a finite state machine as shown in Figure 5.1. The topology of an HMM can be different from the illustration, but the figure shows the typical HMM topology for biological sequence analysis that was introduced in [43].

A path in the HMM begins at the *begin* node and ends at the *end* node. Such a path can only follow the directed edges, and edges never point backwards. This architecture is therefore also called a left to right architecture. A path in the HMM corresponds to an aligned sequence, possibly with gaps and insertions. The generated aligned sequence depends on the nodes traversed by the path. There are three types of nodes in the HMM:

1. Square nodes correspond to matches.

2. Diamond nodes correspond to insertions.

3. Circular nodes correspond to deletions.

Each edge has an associated transition probability, and the square nodes have letter emission probabilities (these probabilities are not shown in the figures). Figures 5.2 and 5.3 show examples of sequences aligned to an HMM using paths from the start node to the end node. Note that given a sequence, there is an exponential number of paths in the HMM and the HMM therefore represents and exponential number of alignments. Each of these paths has an associated probability and it is usually the path/alignment with highest probability that is interesting. One of the applications of HMMs is to generate multiple alignments. If we assume that the aligned sequences shown in Figures 5.2 and 5.3 are high probability paths then the resulting multiple alignment of the sequences is:

```
-A-VPtjC--
KApVA---LK
```

In the example above, extra deletions have been inserted to align the match states (capital letters).

There are advantages and disadvantages of using HMMs for multiple alignments compared to other multiple alignment algorithms. One of the major disadvantages is, that it usually is very time expensive to train the HMM on the appropriate set of sequences. However, this only needs to be done once, and the following alignments of sequences can be done in $O(N^2)$ time compared to other multiple alignment algorithms that are often NP-hard [96].

There are many other applications of HMMs for biological sequence analysis. The application of HMMs that we use for template detection in [64] is the following. HMMs are trained on sets of sequences that are known to be structurally related (a family of proteins). This training adjusts the transition and emission probabilities such that protein sequences of a given family have a high probability in the corresponding HMM. Given a sequence with unknown structure, the path in the HMM with maximum probability can be interpreted as the alignment of the sequence with unknown structure to the family of proteins that the HMM was trained on. The probability of the path indicates if the sequence with unknown structure is a member of the protein family or not.

Training an HMM can to some extent be compared with training a neural network described in Section 4.2. Instead of adjusting the synaptic weights for neural networks, the transition- and emission probabilities are set systematically in accordance with the training set. Several training algorithms have been proposed in the literature and the most used is the expectation minimization algorithm [10].

5.2.2 Target-template Alignment

When the templates have been identified, perhaps using one or more of the algorithms just described, the target is aligned to template. In an alignment

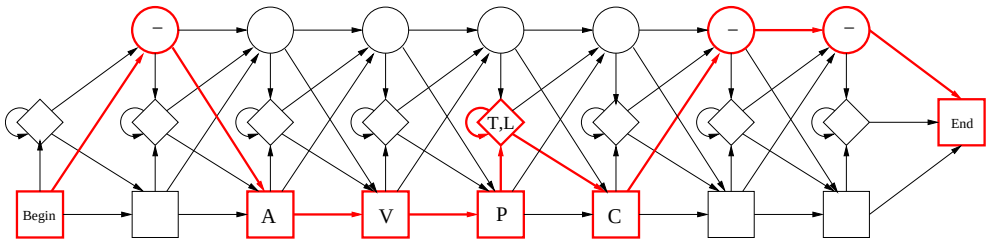


Figure 5.2: The amino acid sequence $AVPTLC$ is aligned to the HMM. The shown path in the HMM generates the alignment $-AVPtlC--$. Capital letters correspond to matches and small letters correspond to insertions.

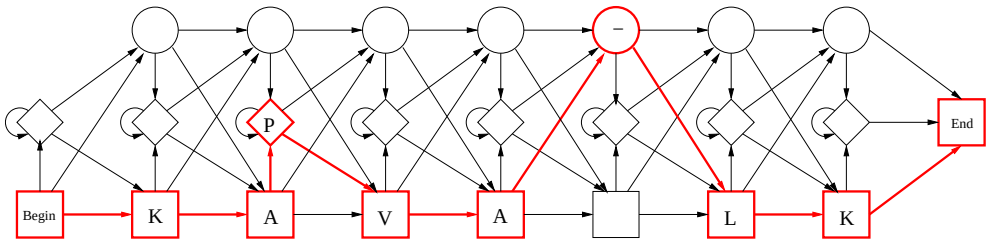


Figure 5.3: The amino acid sequence $KAPVALK$ is aligned to the HMM. The shown path in the HMM generates the alignment $KApVA-LK$.

some of the residues in the target correspond to residues in the template. In most algorithms for template detection, an alignment is also computed. In Figure 5.4 an alignment between a target and a template is shown. Except for exact template matches, such alignments will have deletions and/or insertions in the alignment. If the template and alignment are correct, the matching residues provide a lot of information about the target structure. In Figure 5.5 the C_α-atoms of the matching residues are shown. The coordinates are from the template and thus provide a large amount of information about the global structure of the target.

5.2.3 Model Building

Using the information from the alignments to the templates, the next step is to build full atom models. There as a vast number of approaches to this in the literature. One is to consider short conserved fragments from the templates and assemble them such that some energy function is minimized [87]. Another approach is to extract geometric constraints from the alignments to the templates and find structures that satisfy the constraints [83]. In the model building step, one also often needs to assign coordinates to residues that are not represented in the alignments. This problem is called the *loop closure problem* [11].

The last step in homology modeling is to pick the best of the models generated in the model building step. This is called model quality assessment and is described in more detail in the next chapter.

```
-MVKFACRAITRGRAEGEALVTKEYISFLGGIDKETG-IVKE
m------------ITTGKVWKFGDDISTDEITPGRYNl--TK

DC-E-------IKGESV------------AGRILVFPGGKG-
DPk-elakiaf------ievrpdfarnvrPGDVVVAGKNFGi

-ST--VGSYVLLNLRKNGVAPKAIINKKTETIIAVGAAMAE-
gSSreSAALALKAL---GI---------------AGVIAEs

-------------IPLVEVRDEKFFEAVKTGDRVVVNADEGY
fgrifyrnainigIPLLLGKTEG----LKDGDLVTVNWETGE

V----ELIELEHHHHHH------------------------
VrkgdEILMFEPLE---dflleivreggileyirrrgdlcir
```

Figure 5.4: The blue sequence is the template and the red sequence is the target. The figure shows an alignment between the two sequences. Insertions correspond to small letters, matches correspond to capital letters and deletions correspon to '-'.

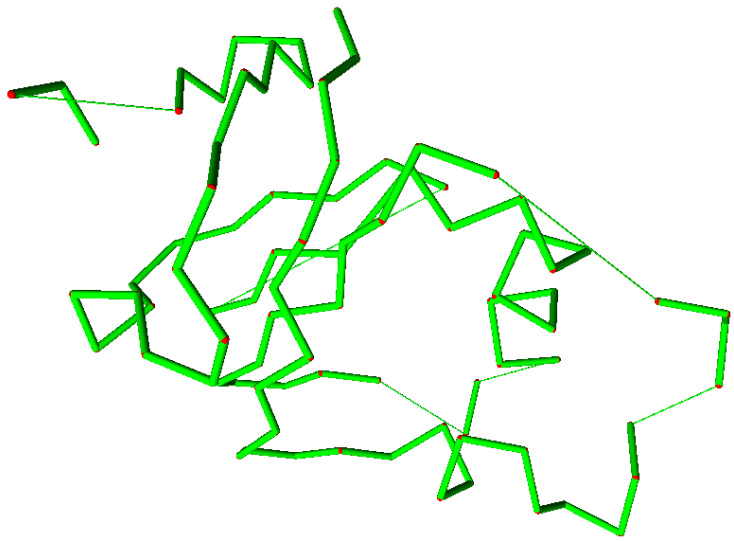


Figure 5.5: The trace of C_α-atoms of the matching residues in Figure 5.4.

5.3 Our Research

Our algorithms for protein structure prediction are based on combinatorial optimization which are described in more details in Chapter 7. However, our algorithm for model quality assessment is based on homology modeling as described in this chapter. Our MQA algorithm does not build models, so the last model building step is omitted. We find templates and alignments using SAM (an HMM). These alignments to templates are used for constructing a set of distance constraints. The distance constraints (between C_β-atoms) are then used in a score function to assess the models in question. Our algorithm for model quality assessment is described in more detail in the next chapter and in [64].

5.4 Chapter Summary

There is a vast number of different approaches for attacking the tertiary structure prediction problem. Some of the basic (non-combinatorial) techniques are molecular dynamics and homology modeling which are described in this chapter. Molecular dynamics is a somewhat naïve approach for computing the atomic trajectories of protein folding. Nevertheless, using molecular dynamics the native structures of small peptides have been predicted. Homology modeling is probably one of the most successful protein structure prediction approaches. Many proteins have one or more so-called homologue counterparts with known structure. A basic idea is therefore to detect these homologue proteins and use them as templates for later model building. In our algorithms for protein structure prediction, we use techniques from combinatorial optimization which are covered in Chapter 7. However, our MQA algorithm is heavily based on techniques from homology modeling.

Chapter 6

Model Quality Assessment

In the previous chapter, it was described how model quality assessment (MQA) often is a natural step in many algorithms for protein structure prediction. MQA algorithms are typically more general and can be used to assess arbitrary models for some target. For example, consider a biologist who sequenced a gene and wants to know the tertiary structure of the corresponding protein. The biologist would of course first query PDB to see if the protein has been analyzed before. If that is not the case, she might want to use a tertiary structure prediction server. There are many online prediction servers available and she might end up using the I-TASSER webserver, because she knows it performed best at the latest CASP7 [98]. On the other hand, the I-TASSER prediction server, does not *always* give the best prediction result among available servers. Another approach is therefore to query many prediction servers known to perform well and select the best *model* generated. This approach, of course, brings up another problem. It is not trivial to determine which model in a set of alternative models is *best* without knowing the native structure. This is the model quality assessment problem. Often, MQA is not only about determining the best model, MQA is often stated as the problem of assigning a score [0:1] to each alternative model of a target, such that the score *correlates* with the *real quality* of the model (Figure 6.1). The *real quality* of the model is usually GDT (as defined in section 3.6), but alternative measures have been used. How the scores should *correlate* with the real quality is not clear and is discussed in more details in the following section.

6.1 Correlation

It is not easy to agree on what measure of quality should be used for evaluating MQA. The reason for this is of course that MQA are used in different contexts. Here we briefly describe three correlation measures; Pearson's r, Spearman's ρ and Kendall's τ. There are other measures of evaluating an MQA, such as the ability to select the best model. Refer to [6] for a description of other MQA evaluation measures.

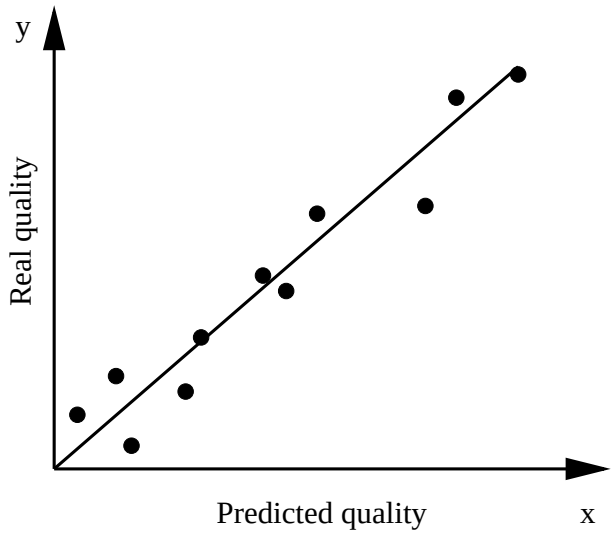


Figure 6.1: Illustration of an MQA with good linear correlation. The points correspond to alternative models for a specific target. The *predicted quality* is the assignment of scores from the MQA algorithm and the *real quality* is the similarity with the model and the native structure (perhaps in terms of GDT). Since the native structure is typically not known when doing MQA, a plot like this can only be made when the native structure is known and the MQA is evaluated.

6.1.1 Pearson's r

When the MQA category was first introduced at CASP7, the MQA algorithms were evaluated using Pearson's r which can be defined as:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}},$$

where (x_i, y_i) corresponds to pairs of (predicted quality, real quality) for all alternative models in the set. $\bar{x}$ and $\bar{y}$ are the average values over all models of x and y correspondingly.

Pearson's r measures the degree of linear correspondence between two variables with a number r in $[-1 : 1]$. Pearson's r would therefore be high on the linearily correlated points shown in Figure 6.1. In our MQA paper [64] we claim that Pearson's r is inappropriate for evaluation of MQA, because we generally do not care about the *linearity* of the MQA prediction. An example of what we think could be a perfect MQA is shown in Figure 6.2. Even though the MQA in this ad-hoc example is able to pin-point the best model and perfectly rank all models, it has a low Pearson's r because the points are not linearily correlated. To avoid this inappropriate evaluation, we therefore propose to use Spearman's ρ or even better, Kendall's τ as correlation measure for MQA.

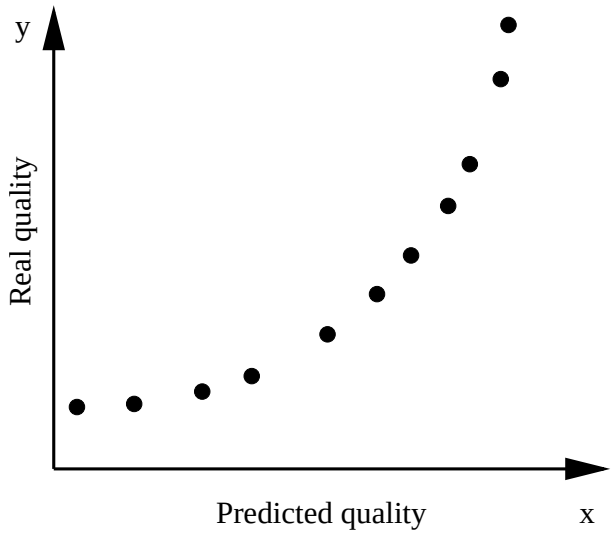


Figure 6.2: Illustration of an artificial MQA with a low Pearson's r.

6.1.2 Spearman's ρ

Spearman's ρ is a special case of Pearson's r. When computing Spearman's ρ, the raw data is first converted to ranks and then Pearson's r is computed for the ranks. Spearman's ρ is therefore maximum for the perfect correlated points in Figure 6.2.

6.1.3 Kendall's τ

Kendall's τ also measures the correspondence between two rankings and is defined as

$$\tau = \frac{4P}{n(n-1)} - 1,$$

where n is the number of points and P the number of concordant pairs. A pair of points is said to be concordant if

$$\text{sign}(X_A - X_B) = \text{sign}(Y_A - Y_B)$$

If two random points (A and B) are chosen and $X_A > X_B$ then Kendall's τ is proportional to the probability that $Y_A > Y_B$. We prefer Kendall's τ over Spearman's ρ, because it is more interpretable, and in our paper [64] we show examples where Kendall's τ agrees more with our intuition of a good MQA than Pearson's r and Spearman's ρ.

6.2 Algorithms for MQA

The ability to assess the quality of a protein model is a fundamental problem in the field of protein structure prediction and many different algorithms have been described in the literature. Recently the CASP organizers recognized the

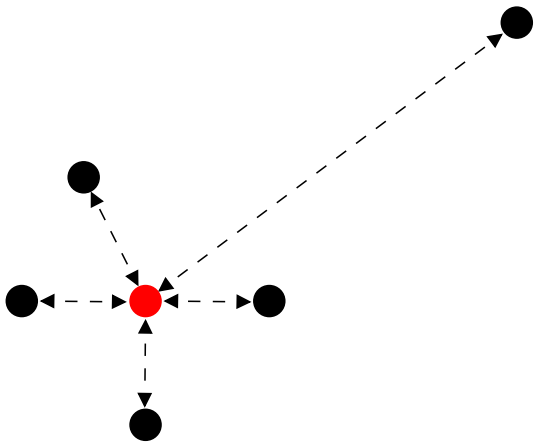


Figure 6.3: An illustration of 6 models where 5 of the models are clustered (according to some arbitrary metric). The red model is the highest scoring model, because it has the lowest mean distance to the other models.

importance of MQA and therefore introduced the MQA category in CASP7 where 26 groups participated. Two algorithms (Pcons and Lee) showed to be superior to the rest of the groups and we briefly describe these two algorithms here. We also describe two new MQA algorithm (that did not participate in CASP7). One is based on *support vector regression* (SVR) and the other uses a new weight optimization algorithm. Finally we briefly introduce our algorithm for MQA, which is described in more details in [64].

6.2.1 Pcons

Pcons [94, 95] is a consensus algorithm that measures the similarity of each model to the other models. Pcons uses LGscore [18] as a similarity measure, but any similarity measure can in principle be applied. The score of a model therefore corresponds to the average similarity between the model and the other models in the set. A model that is very similar to many other models in the set would therefore score high. Any consensus algorithm, like Pcons, depends on the quality of the input set of the models. Even if the input set *does* contain a very good model, the consensus approach might fail if the input set also have a large number of bad and structural similar models. When assessing models from good automated prediction servers (like in CASP7 MQA), the input set often contains many good models which makes consensus approaches appropriate.

6.2.2 Lee's Algorithm

The second best CASP7 MQA algorithm was the Lee algorithm. The basic idea in Lee's algorithm is very simple. First a tertiary structure prediction of the target is made (The Lee group of course use their own prediction algorithm). Then the similarity between each model in the set and Lee's prediction is measured and the models are scored accordingly (Figure 6.4). If the tertiary structure

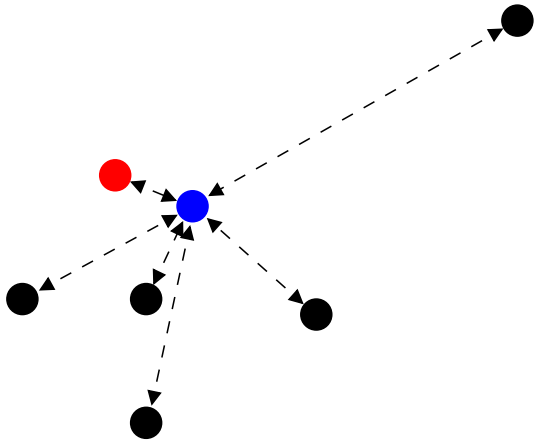


Figure 6.4: An illustration of 6 models to be assessed and one predicted model (blue) by Lee's tertiary structure prediction algorithm. The red model is closest to the predicted model (in terms of GDT) and is therefore given the highest score.

prediction is good (which is often the case with Lee's predictions) this approach is of course successful. In the opposite case, Lee's algorithm is known to produce very bad MQAs when their tertiary structure prediction is wrong [64].

6.2.3 Support Vector Regression

An example of a new MQA algorithm that did not participate in CASP7 is the support vector regression algorithm by Qiu et al. [79]. The idea is to consider many features that somehow describe the quality of the models. The SVR algorithm by Qiu et al., considers a total of 25 features divided in two categories; 4 consensus based features and 21 structural features. The consensus based features include a score function similar to the Pcons approach and the structural features are computed from the individual models (i.e. score functions based on pairwise atomic interactions, hydrophobic packing, angle preferences etc.). The purpose of the algorithm, is to end up with a linear function of these features such that it approximates the GDT of the models:

$$\mathrm{GDT_a}(x) = w_1 f_1(x) + w_2 f_2(x) + \cdots + a_n f_n(x) + b$$

In the equation above, $\mathrm{GDT_a}$ is the approximated GDT computed by the function. The feature functions f_i, $1 \leq i \leq n$, depend on the model x and $\bar{w}$ and b are parameters that must be set appropriately. A good linear function therefore minimizes the error between the real GDT ($\mathrm{GDT_r}$) and $\mathrm{GDT_a}$. To accomplish this, the weights are adjusted using the machine learning technique, SVR. When treating the problem as an SVR problem, a training set is used for solving the convex optimization problem:

$$\text{Minimize} \qquad \tfrac{1}{2}||w||^2 \tag{6.1}$$

$$\begin{aligned} \text{Subject to} \quad & \mathrm{GDT_r}(i) - \mathrm{GDT_a}(i) - b \leq \epsilon && (6.2)\\ & \mathrm{GDT_a}(i) - \mathrm{GDT_r}(i) + b \leq \epsilon && (6.3)\\ & \forall i = 1, 2, \cdots, n && (6.4) \end{aligned}$$

Where w are the weights of the features. $GDT_a(i)$ is the approximate GDT of the i'th training example and $GDT_r(i)$ is the real GDT of the i'th training example. When solving the problem stated above, we find a solution where the errors are within the predefined range ϵ and the sum of the squared weights is minimal. In practice, however, it is inappropriate to predefine ϵ. If ϵ is too small, the problem might not contain any feasible solutions and if ϵ is too large, the function might generate many errors. A more useful alternative formulation used by Qiu et al. is therefore:

$$\begin{aligned} \text{Minimize} \quad & \tfrac{1}{2}||w||^2 + \textstyle\sum_{i=1}^{n} C_i(\zeta_i + \hat{\zeta}_i) && (6.5)\\ \text{Subject to} \quad & \mathrm{GDT_r}(i) - \mathrm{GDT_a}(i) - b \leq \epsilon + \zeta_i && (6.6)\\ & \mathrm{GDT_a}(i) - \mathrm{GDT_r}(i) + b \leq \epsilon + \hat{\zeta}_i && (6.7)\\ & \zeta_i, \hat{\zeta}_i \geq 0 \qquad \forall i = 1, 2, \cdots, n && (6.8) \end{aligned}$$

Where ζ and $\hat{\zeta}$ are variables that make sure that a feasible solution always exists. The constants C_i, $1 \leq i \leq n$, are predefined and correspond to a trade-off between the weight minimization and the error minimization. In the implementation by Qiu et al., they use a higher weight on high ranked models because they want the algorithm to perform better on good models. The SVR algorithm is illustrated in Figure 6.5.

Not surprisingly, after solving the optimization problem, it turns out to be a consensus feature that is given the highest weight. Qui et al. claim that their MQA algorithm outperforms all MQA algorithms at CASP7.

6.2.4 Weight Optimization

Other algorithms for learning the weights of a linear function of features have been proposed in literature. Here, the weight optimization approach from Archie et al. [6] is briefly described. This algorithm is interesting in this study, since some of the features are alignment constraints from our MQA algorithm [64] described in the next section. The optimization algorithm consists of a number of so-called *rebalancing* steps. The basic idea is to divide the features in two sets $(f_1, \cdots, f_m)$ and $(f_{m+1}, \cdots, f_z)$ and let the cost function depend on a parameter $(0 \leq p \leq 1)$ such that:

$$\mathrm{Cost}(x) = p(w_1 f_1(x) + \cdots + a_m f_m(x)) + (1-p)(w_{m+1} f_{m+1}(x) + \cdots + a_z f_z(x))$$

For fixed weights, the idea is to find a value of p that gives the *best* cost for some training set. The best costs in this context are values that correlate well with GDT. The parameter p that optimizes the correlation between Cost(x) and GDT in the above equation is determined using Brent's method [76]. When p is determined, the weights $w_1, \cdots, w_m$ are multiplied by p and the weights

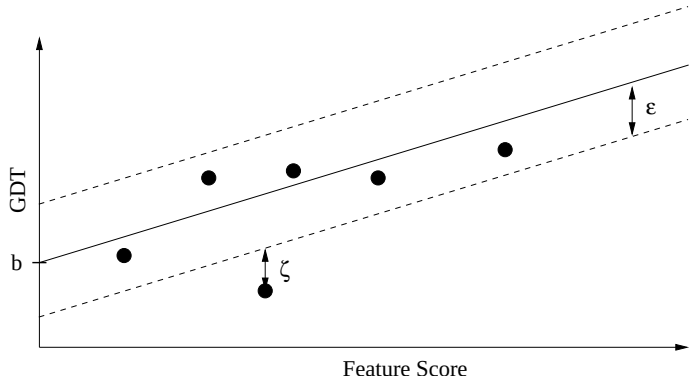


Figure 6.5: For illustration purposes, only one feature is considered here. The slope of the solid line therefore corresponds to the weight of the feature, and the y-axis intersection corresponds to the b parameter. In this example all but one model are within the ϵ range of the line shown and a feasible solution to the problem in Equation 6.1 therefore does not exist (for this slope and b-value). When solving the alternative formulation in Equation 6.5, the ζ of the outlying model is positive and the solution becomes feasible.

$w_{m+1}, \cdots, w_z$ are multiplied by $(p-1)$. This optimization algorithm begins with an initialization of the parameters (see [6] for details) and continues with a number of the rebalancing steps until no improvements in correlation can be found.

When no consensus features are used, this MQA algorithm performs slightly better than the other MQA algorithms described in this chapter. In this case, the most significant features are the alignment constraints (Section 6.3). When model consensus features are added, the MQA algorithm performs significantly better than all other MQA algorithms, and the most significant features, of course, become the consensus based features.

6.3 Our Research

We have developed an algorithm for MQA and tested it on the CASP7 benchmark. The algorithm is described in details in [64] and an overview is briefly described here.

6.3.1 Overview

There are 5 main steps in the MQA algorithm which are also illustrated in Figure 6.6:

a. The input to the MQA algorithm is the amino acid sequence of the target and a set of alternative models for the target.

b. We use SAM_T06 for detecting homologues and computing the alignments. SAM_T06 also returns an E-value for each template found. Tem-

plates with low E-value are on average more correct than templates with high E-value.

c. SAM_T06 also computes the alignments to the templates. For each residue pair with chain separation greater than or equal to 9, we store the distance between C_β-atoms from the alignments if the distance is less than 8 Å. We therefore end up with a length × length - table with sets of observed distances between C_β-atoms as illustrated in Figure 6.7. Then a weighted average for all entries in the residue × residue table is computed. The weights associated with each weight depend on the E-value as described in [64]. The resulting table (Figure 6.8) contains a so-called *desired distance* and a *confidence-value* (weight) of the desired distance. This weight is in the interval [0:1]. If a pair of C_β-atoms has been in contact in many high quality alignments, the weight of the constraint is high (near 1) and if the pair of C_β-atoms has been in contact in few low quality alignments the weight is low (near 0).

d. Each entry in the table generates a so-called *distance constraint*. If the entry has a desired distance, the distance constraint is a function with minimum value in the desired distance as shown in Figure 6.9. Otherwise, the entry generates a so-called non-contact as shown in Figure 6.10.

e. The final model cost function is the weighted sum of all distance constraints. Given a model, the distance between each pair of C_β-atoms is measured and the corresponding distance constraint value is computed. The weighted sum of all distance constraint values is the cost of the model. We rescale the costs such that they correspond to scores in the interval [0:1] where 1 corresponds to the best scoring model and 0 is the worst scoring model.

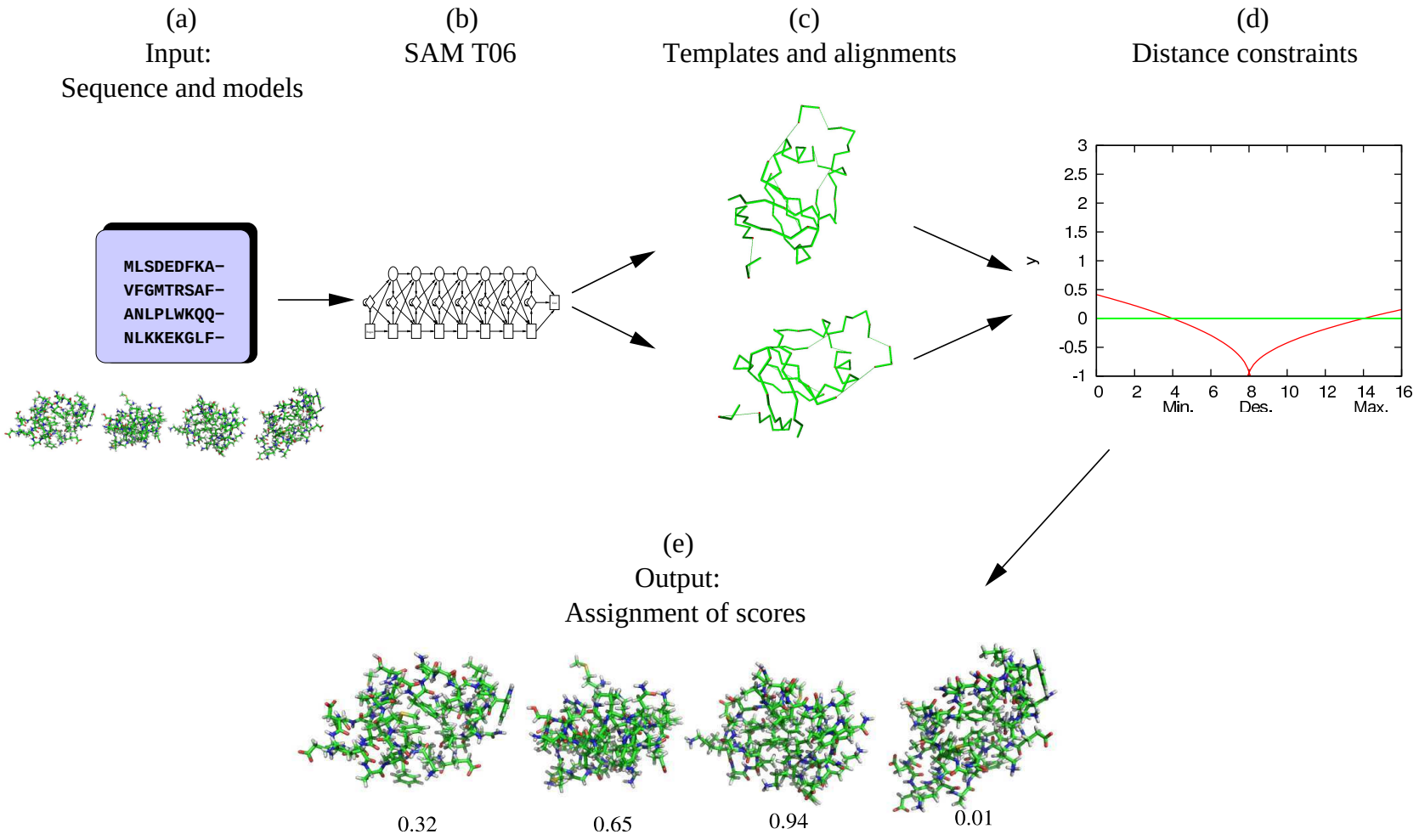


Figure 6.6: Overview of our MQA algorithm.

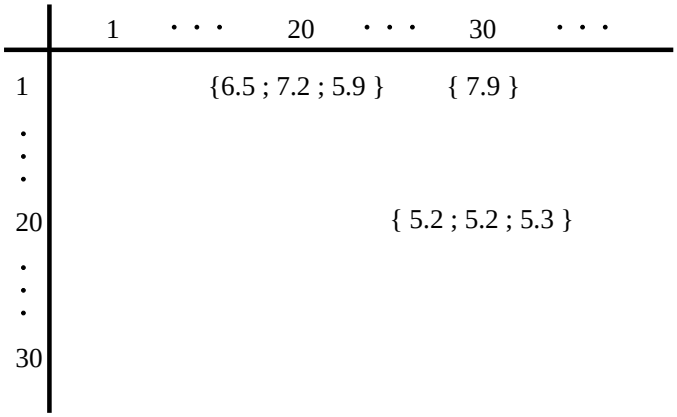


Figure 6.7: A table of observed distances (≤ 8) between pairs of C_β-atoms is constructed.

6.3.2 Optimization

Figure 6.11(a, c, e) shows examples of the quality of the distance constraints for three targets. Target T0314 is known to be one of the most difficult targets of CASP7 in terms of prediction quality. The main reason is that no good homologues have been detected. T0365 is template-based and considered to be medium difficult. T0346 is a so-called high-accuracy template-based target and is the easiest target of CASP7. At least for the template based targets, the figure shows a clear correspondence between constraint weight and constraint quality. It is therefore an obvious improvement strategy to select and use only the high weight constraints. In [64] we describe two selection strategies. One is to select and use only the high weight constraints. When evaluating our MQA algorithm using this approach, the performance is slightly improved compared to using all constraints. However, the most useful selection strategy we have tested is an optimization technique based on contact number probability distributions. We use the feed-forward neural network called *predict-2nd* [38] to predict the probability of the residue having various numbers of contacts. Figure 6.13 illustrates an example of such a prediction of a contact number distribution. One of the objectives in the optimization approach is therefore to select a subset of the constraints such that the total contact probability is maximized. The other objective is to maximize the average weight of the constraints selected. We use a simple greedy approach to find solutions to this problem as described in more details in [64]. The consequence of selecting the constraints using the optimization approach is illustrated in Figures 6.11 (b,d,f) and 6.12 (b,d,f). It is clear that we are able to filter away many of the wrong constraints, which eventually leads to better correlations of MQA.

6.3.3 Evaluation

We have compared our MQA algorithm with other MQA algorithms in the literature, including the two best ranked algorithms at CASP7 and the MQA

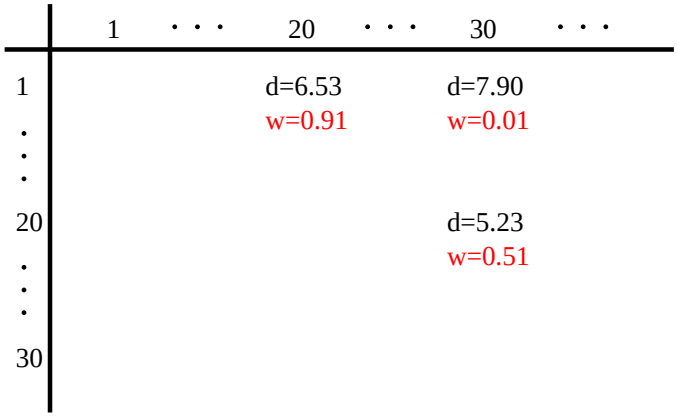


Figure 6.8: A weighted average distance is computed together with the confidence-weight of the distance. In this ad-hoc example there is a high confidence that the C_β-atoms of residue 1 and residue 20 are near 6.53 Å from each other.

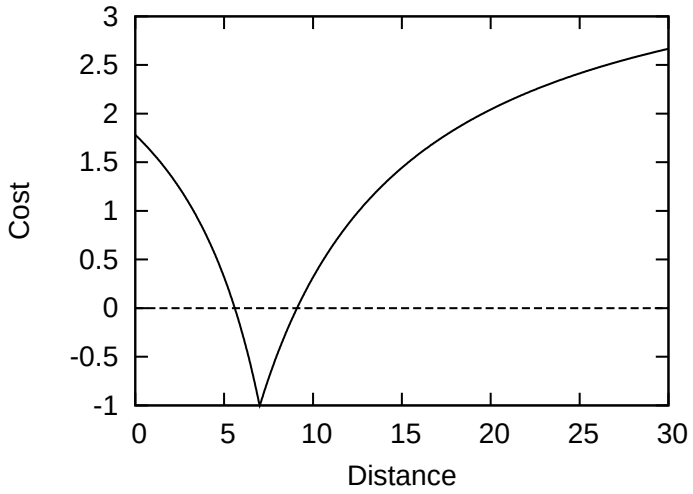


Figure 6.9: The cost function of the distance constraint where the desired distance is 7 Å.

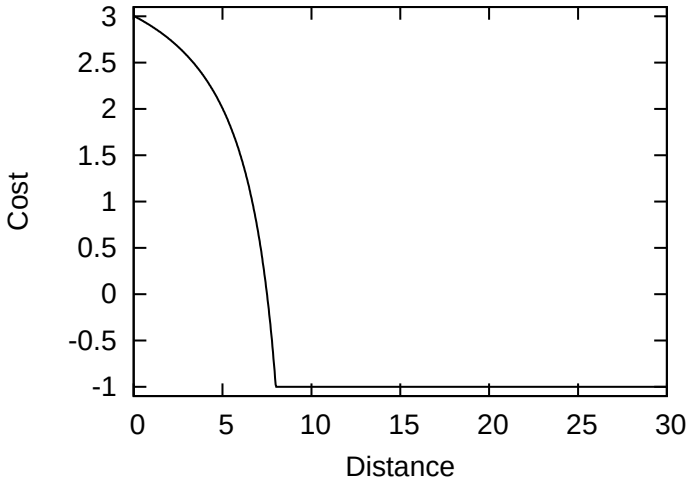


Figure 6.10: A plot of the non-contact cost function. These cost functions correspond to pairs of residues where contacts between C_β atoms have not been observed in any alignment.

algorithm by Qiu et al. Table 6.1 shows the results of this comparison (refer to [64] for details). The *Constraints (Consensus)*-row in the table is our MQA algorithm when the distance constraints are extracted from the models instead of alignments. This corresponds to a model consensus approach which again shows the best performance on the CASP7 MQA benchmark. The *Undertaker* row is from the weight optimization algorithm described in Section 6.2.4, where 73 different features from the Undertaker protein structure prediction program are used. Among these features are the alignment constraints which proves to be the most significant of the features.

6.4 Chapter Summary

Model Quality Assessment (MQA) is the problem of assigning a quality measure to alternative models of a target without knowing the native structure. It is a natural step in many algorithms for protein structure prediction and other applications. The MQA category has recently been presented at CASP7. It is not a trivial task to evaluate an MQA algorithm. Several correlation methods have been proposed and we argue that Kendall's τ is one of the most appropriate measures for evaluating MQA algorithms. The best MQA algorithm at CASP7 was a consensus based algorithm that scored the models according to their mean distance to other models (in terms of LGscore). Consensus based algorithms require a set of good models (to derive consensus from) and can therefore not be used for assessing few models. In the extreme case where the quality of one or two models should be assessed, it does not make sense to use a consensus approach. Our approach for MQA does not have this requirement. The score function we use, is based on distance constraints from alignments. Our MQA algorithm therefore performs best on template based targets. We also show how

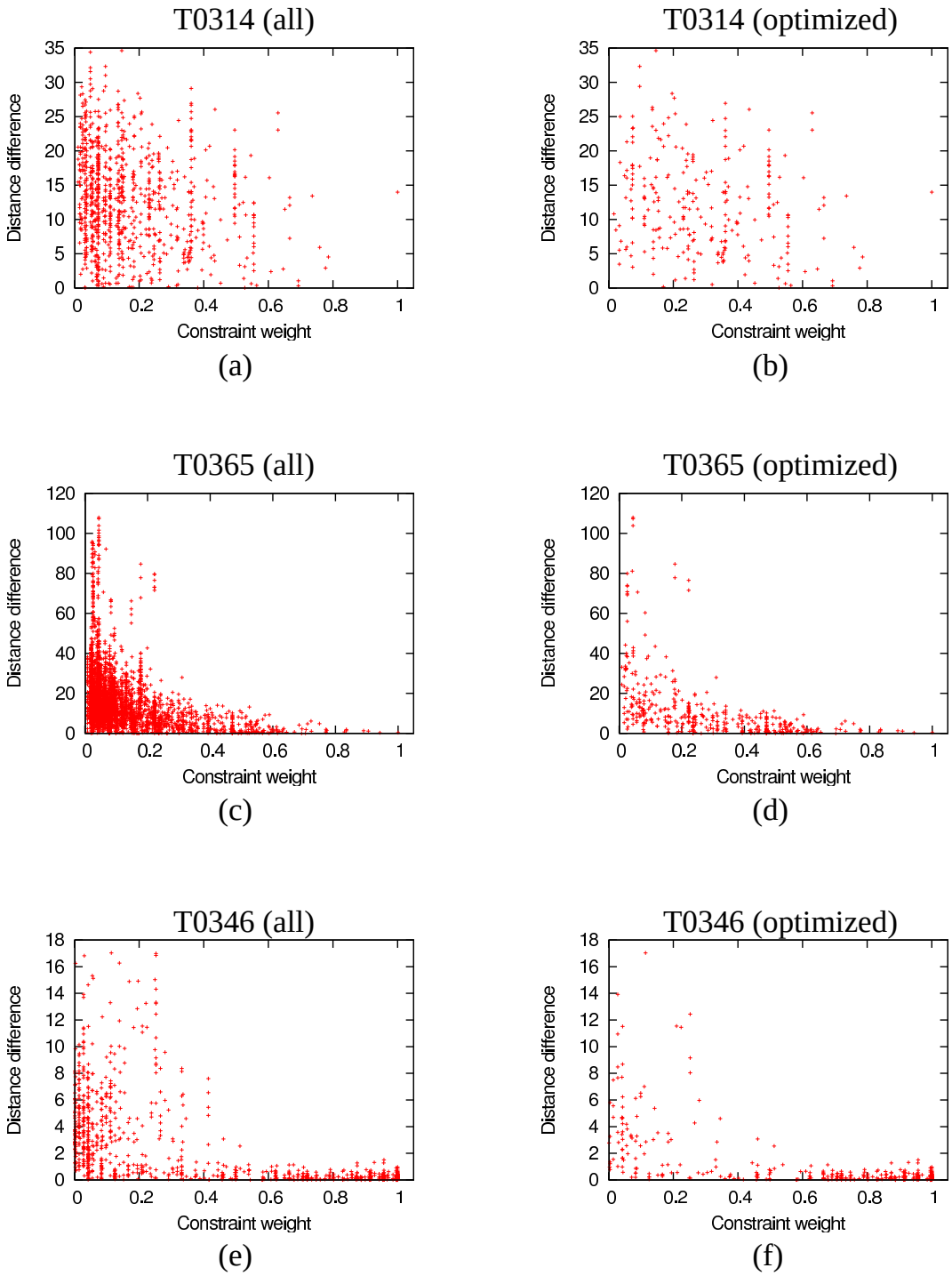


Figure 6.11: Quality of distance constraints for three targets ranging from very difficult (T0314) to very easy (T0346). The *distance difference* is the absolute value of the difference between the *desired distance* of the constraint and the *real distance* in the native structure.

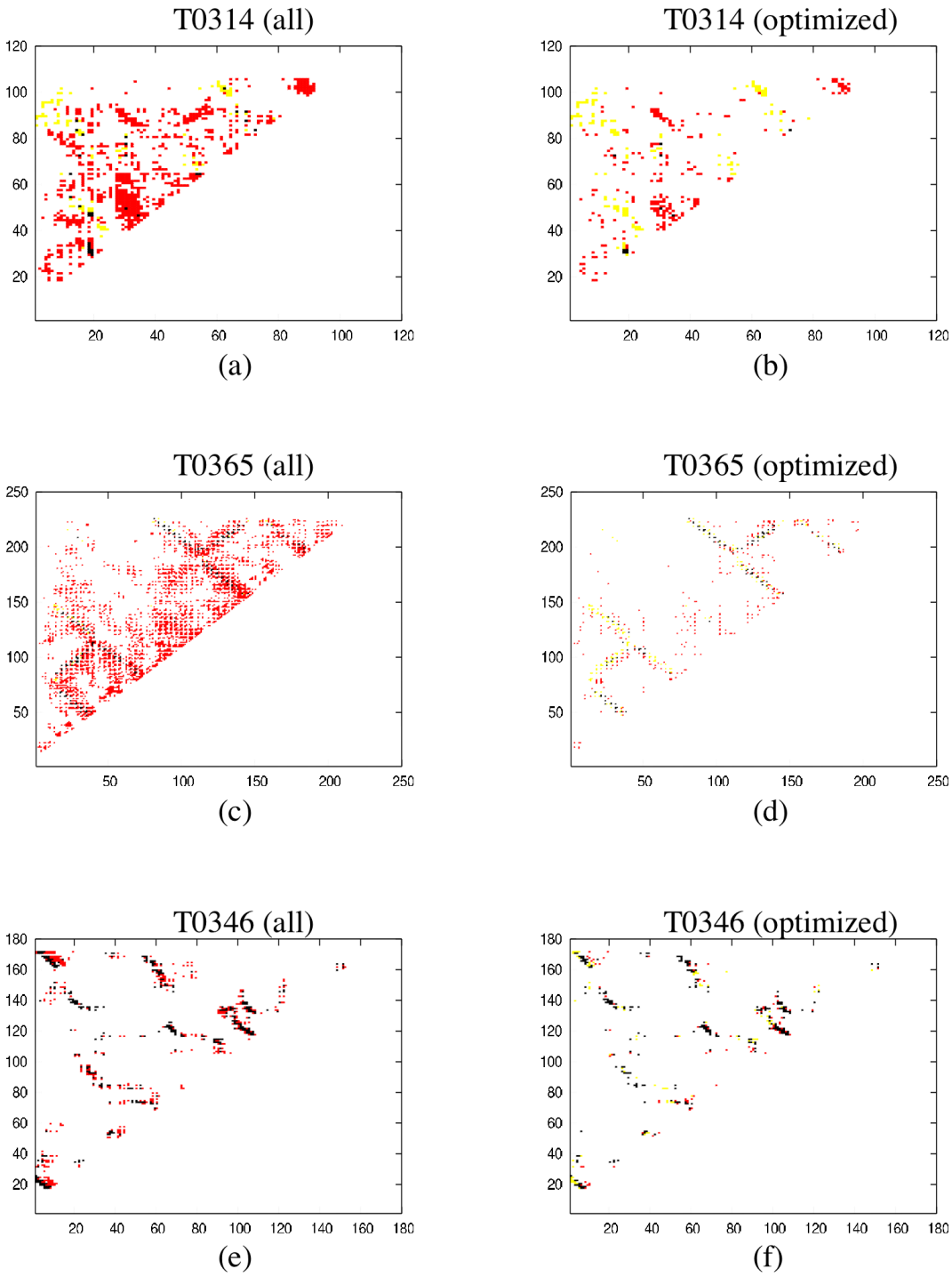


Figure 6.12: Contact maps for three targets ranging from very difficult (T0314) to very easy (T0346). Black points correspond to contacts in the native structure that are correctly predicted (a distance constraint is generated). Yellow points correspond to contacts in the native structure that we missed (no distance constraint generated) and red points correspond to pairs of residues not in contact but having a distance constraint. White points corresponds to correctly predicted non-contacts. From the figures, it is clear that mainly the red points are being filtered away by the optimzation algorithm, while leaving the black points.

Group	$\bar{\tau}$	$\bar{r}$
Constraints (Consensus)	**0.62**	**0.86**
Undertaker (with align. constr.)	**0.62**	**0.86**
Lee	0.59	0.81
Qiu	0.58	0.85
Constraints (Aligments)	0.57	0.83
Pcons	0.56	0.85

Table 6.1: The table shows the average Kendall's τ and average Pearson's r for MQA with each algorithm compared, ranked using Kendall's τ. The average values are on a per-target basis. The *Constraints (Alignments)* row is the results of MQA with distance constraints from alignments. The *Constraints (Consensus)* row is the results of extracting the constraints from the models to be assessed. The *Undertaker* row is the results of using all Undertaker cost functions including the alignment constraint sets. Lee, and Pcons are top ranked MQA algorithms presented at CASP7 (groups 556 and 634 respectively). Qiu is the SVR MQA algorithm. In this table only the full backbone models are evaluated. Our distance constraints perform worse on models with missing backbone atoms, because a subset of the distance constraints can not be evaluated. In [6] the Undertaker constraints (together with the alignment constraints) are also evaluated on all models including the broken models.

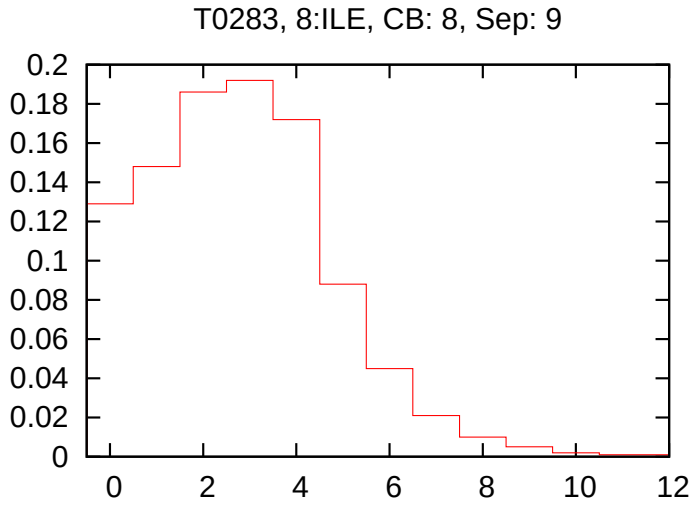


Figure 6.13: A contact number probability distribution for residue number 8 (isoleucine) of target T0283. The plot shows that there is maximum probability that the C_β-atom of the residue is in contact (≤ 8) with three other C_β-atoms (with chain-separation 9 or more). The probability of 10 or more contacts for this residue is almost zero.

to apply contact number predictions for selecting good distance constraints.

Chapter 7

Structure Prediction using Combinatorial Optimization

The protein structure prediction problem can be treated as a standard combinatorial optimization problem which is one of the classical disciplines in computer science. A combinatorial optimization problem consists of a mathematical object which has different discrete states. Each of these states has an associated value which is defined by a so-called objective function. The solution to the combinatorial optimization problem is the state with the global minimum (or maximum) value of the objective function. When treating the protein structure prediction problem as a combinatorial optimization problem, we therefore need to discretize the different structures of the polypeptide chain and associate an objective value to every state. The objective value should somehow represent the real Gibbs free energy of the polypeptide chain. However, it is not trivial to discretize the polypeptide chain in a reasonable manner. In nature there is an infinite number of possible structures of a polypeptide chain, so any discretization is more or less unnatural. When we treat the protein structure prediction problem as a combinatorial optimization problem, we therefore sacrifice some realism to achieve computational tractability.

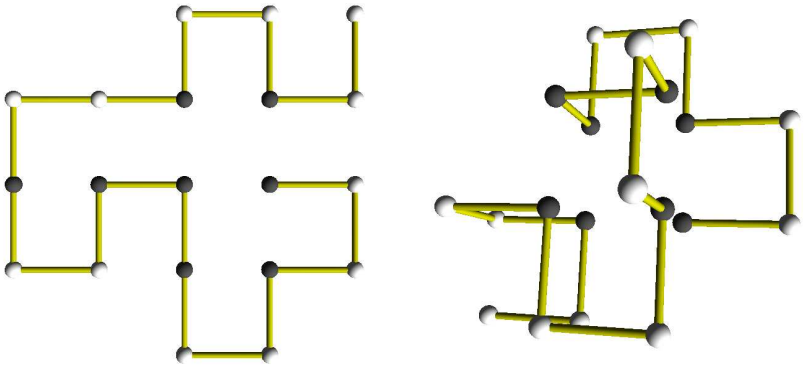
7.1 Discrete Representations of a Polypeptide Chain

A straightforward discretization of the polypeptide chain is to treat the amid planes as rigid objects and only allow discrete values of the ϕ and ψ angles (shown in Figure 2.5 page 17). Note that even a very rough discretization of 4 different values of the dihedral angles gives an astronomical number of possible structures. The number N of possible structures using this discretization (not counting clashing structures or other unnatural structures) is

$$N(L) = 4^{2L-2}$$

where L is the number of amino acids in the polypeptide chain. The number of conformations for a protein of 200 amino acids therefore is:

$$N(200) = 4^{398} \simeq 4 * 10^{239}$$



(a) (b)

Figure 7.1: (A) The string of beads on a 2D quadratic lattice. (B) The string of beads on a 3D cubic lattice. The 2D lattice representation is of course very unnatural because no protein is 2D, however, this lattice can be useful for evaluation and analysis of algorithms. The 3D lattice representation is more natural but still considered to be a very rough discretization.

so, even with this very rough discretization, the number of possible structures are many times bigger than the number of atoms in the universe - and we have not even considered the degree of freedom of the side chains.

7.1.1 Discretization using Lattices

One of the much simpler representations is to only consider one atom per residue. This representation is often called *a string of beads*. Each bead typically represents either the C_α-atom or C_β-atom of the amino acids. Discretization of the string of beads can be done in many ways. One of the widely used discretizations is to force each bead to be positioned on a lattice (Figure 7.1).

The advantages of using a lattice for the discretization of a string of beads are many:

- There is a finite number of structures, and complete enumeration is possible for small chains.
- Comparison of structures is easy. Structures are different if they have a different path in the lattice and it is easy to check if two structures are rotational identical.
- It is easier to do exact computations and rounding errors can often be eliminated by considering lattice coordinates instead of space coordinates.
- Many algorithmic problems can be solved very efficiently. Collision detection occurs only when beads occupy the same lattice node. Finding

the neighbours of a bead can be done by only searching the neighbouring lattice nodes. Local moves can be computed very fast.

The only important disadvantage of using a lattice to represent the chain of beads is that real proteins in nature are *not* represented in lattices. Lattice models therefore only approximate real proteins to some extent. However, in section 7.5.1, we show how more complex lattices can reduce this problem.

7.1.2 The HP-model

In 1989 Lau and Dill presented the well-known HP-model [48]. In their model the string of beads corresponds to a polypeptide chain where each residue is classified in the two categories; hydrophobic nonpolar (H) or hydrophilic polar (P). Hydrophobicity is one of the important properties of amino acids and was introduced in Section 3.3 page 26. In the original HP-model, the string of hydrophobic and hydrophilic beads is only allowed to occupy 2D-cubic lattice nodes and the score-function is the number of hydrophobic (H) neighbours in the lattice. Using complete enumeration they find the structure(s) with maximum number of H-neighbours and they argue that these structures share some properties with real proteins. Structures with maximum number of H-neighbours tend to have a high compactness and a hydrophobic core like water soluble proteins. The 2D-structure in Figure 7.1 is an example of a solutions in the HP-model where the black nodes correspond to hydrophobic amino acids and the white nodes correspond to hydrophilic amino acids. Lau and Dill did not use any optimization technique other than complete enumeration when they presented their algorithm in 1989. In the next section it is described how to find good or even optimal solutions when complete enumeration is not feasible.

7.2 Solving Combinatorial Optimization Problems

Many real-world problems can be treated as combinatorial optimization problems. Typical examples are the traveling salesman problem [45], vehicle routing problem [45] and knapsack problem [40]. These examples are all known to be NP-hard which is often the case for combinatorial optimization problems. Hart and Istrail [30] also showed that finding optimal structures in one of the simplest formulations of the protein structure prediction problem, the 2D HP-model, is NP-hard. When dealing with NP-hard problems, people generally use three different approaches.

a. **Exact algorithms.** Even though a problem is NP-hard, it might be possible to solve realistic problem instances in reasonable time. For some problems, much is known about the structure of optimal solutions, which can be used for constructing efficient exact algorithms. This is the case for the HP-model. Even though it is NP-hard, Backofen et al. [7] are able to compute fast exact solutions to instances with up to 200 residues in the HP-model using their theory of compact hydrophobic cores.

b. **Approximation algorithms.** An approximation algorithm runs in polynomial time and is able to give a guarantee on the quality of the solution. Not all NP-hard problems can be approximated, Hart and Istrail [30], however, showed that the HP-model can be approximated with a guaranteed energy within 3/8 of optimal energy.

c. **Heuristics.** Heuristic search algorithms do not give a guarantee on the solution quality. Nevertheless, in practice, heuristic algorithms are preferred for many combinatorial optimization problems. This is mainly because they are easy to implement and imperically show good solution qualities.

7.3 Metaheuristics for Protein Structure Prediction

In the following sections, different popular metaheuristics are described. Metaheuristics are heuristics that are more general and can be applied to a broad range of optimization problems. The metaheuristics included here (Monte Carlo Search, Tabu search and Bee colony optimization) are just a small subset of metaheuristics proposed in the literature. They are chosen because we use them in our research described in Section 7.5.

7.3.1 Monte Carlo Search

In protein structure prediction, the metaheuristic Metropolis Monte Carlo Search [58] (here we just name it Monte Carlo Search) is one of the most applied metaheuristics. This can be explained by the simplicity of the metaheuristic and the similarity with physical systems. Monte Carlo-based search algorithms differ from molecular dynamics algorithms by being nondeterministic. For these algorithms *randomness* is important and that is why they are named after the famous casino in Monaco. There are many variants of Monte Carlo search (simulated annealing [42], replica exchange [90], Markov chain Monte Carlo [3], etc.). The standard approach is to maintain a single current structure and iteratively apply small random changes to it in each iteration. If the energy of the modified structure is lower than the current structure, the modified structure is automatically accepted as the current structure. If the energy of the modified structure is higher than the current structure, the modified structure is accepted with some probability. The standard probability of accepting a modified structure s', given the current structure s is

$$P(s'|s) = e^{-\frac{U(s')-U(s)}{T}}$$

where U is the energy function and T is the temperature. For high temperatures, almost all solutions are accepted and for low temperatures almost only the improving solutions are accepted. When applying the MC metaheuristic, it is important to determine a suitable temperature. A widely used variant of MC is called *simulated annealing* (SA) where the temperature is gradually decreased during the SA run.

The small changes applied to a structure defines a *neighbourhood* of structures. As the name indicates, neighbouring structures should be close in the

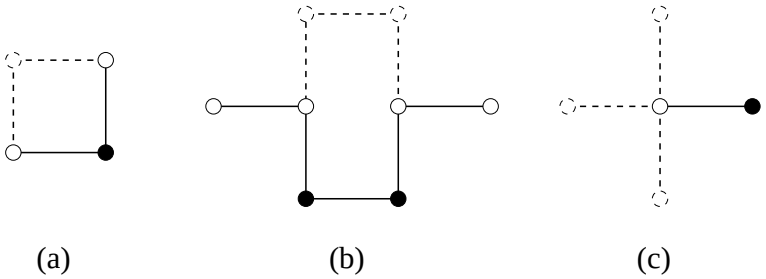


Figure 7.2: Three examples of moves in a quadratic lattice. The black bead is being moved into the position of the dashed white bead. The positions of other solid white beads are fixed. (a) Corner move. (b) Crankshaft move. (c) The end move (three different moves only allowed by the two end beads). These moves are 2D version of the move set by Sali et al. [84]

solution space and therefore share some properties. A neighbourhood is often defined by a *move set*. In Figure 7.2 an example of a very simple move set for the string of beads on a 2D cubic lattice is illustrated.

7.3.2 Tabu Search

One of the most successful metaheuristic in combinatorial optimization is *tabu search* (TS). It has shown to be successful for many applications like vehicle routing [25], VLSI routing [89, 53], packing problems [77] etc., but it has not been given much attention in the field of protein structure prediction. The search paradigms presented so far (molecular dynamics (MD) in Section 5.1 page 43 and MC) have roots in physics. However, many models for protein structure prediction are not based on physics. They are often extremely simplified and discretized - and their energy functions might not even contain any physical-derived terms. In these cases, it is therefore not likely that MD or MC could simulate the real folding pathways. In most cases the only interesting structures are those with low energy. Like MD and MC, tabu search do provide a pathway of examined solutions, however it should not be given a physical interpretation.

TS was first described by Fred Glover in 1989 [26]. It is a local search algorithm with a memory. A local search algorithm iteratively chooses some solution that improves the current solution. At some point it therefore ends in a local optimum where no neighbourhood solution can improve the current solution. This local optimum might also be global, but this can usually not be determined. The risk of getting trapped in local minima can be reduced using a memory. In the most simple implementations of TS, the memory consists of previously visited solutions stored in a so-called tabu-list. When choosing the new neighbourhood solution, the memory is scanned to make sure that it has not been visited before. This simple tabu definition was used by Oakley et al. [61] for prediction of protein aggregation with modest success. Tabu search can also be used for marking regions of the search space as tabu. This is often done by defining some of the attributes of visited solutions tabu. In most TS

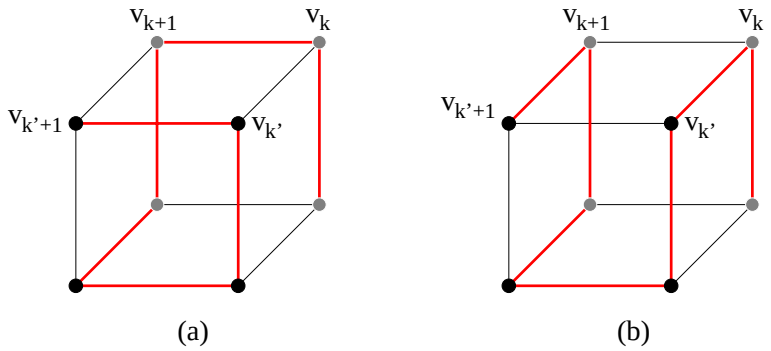


Figure 7.3: (a) shows a path where the conditions for an edge patch is present. (b) shows the result of an edge patch.

algorithms the tabu list is considered to be short term memory and is therefore implemented using a FIFO queue with a predefined length.

One of the few applications of TS on protein structure prediction is the study by Pardalos et al. [69]. They use a cubic lattice to represent the string of beads as a self-avoiding path. The objective function is a simple statistically derived potential energy. The neighbourhood of structures consists of the so-called *edge patches*. An edge patch is a global change (compared to the local moves illustrated in Figure 7.2) to the self-avoiding path in the lattice and it can be made when the following conditions are present:

Let $v_1, v_2, \cdots, v_n$ be lattice nodes visited by the string of beads in this order. If $k < k'$ and $v_k, v_{k'}$ are neighbours and v_{k+1} and $v_{k'+1}$ are also neighbours. Then a new path in the lattice can be generated by adding the two edges (v_k, $v_{k'}$) and (v_{k+1}, $v_{k'+1}$), and deleting the two edges (v_k, v_{k+1}) and ($v_{k'}, v_{k'+1}$). An example of an edge patch is shown in Figure 7.3.

An edge patch is characterized by the edges that are inserted and the edges that are removed. In the TS algorithm by Pardalos et al., the applied edge patches are inserted in the tabu list and future moves are prevented from applying previously used edge patches. Using this strategy, a region of the conformational space is made tabu with each edge patch in the tabu list. The advantage of marking regions tabu compared to just making previously visited solutions tabu, is that the search escapes local minima faster. However, we also face the risk that good solutions are made tabu just because they share some properties with bad solutions. The edge patch move just described can make very large changes to the structure. The risk is therefore, that a structure is defined as tabu, even though it structurally does not have any similarities with a previously visited structure. This risk is reduced by the *aspiration criteria*; if a solution in the neighbourhood is better than the best observed solution, it is accepted even if it is tabu.

In our paper *Reconstructing Protein Structure from Solvent Exposure using Tabu Search* [63] we minimize the risk of making good structures tabu by introducing a new tabu definition. It is directly based on structural differences as described in more detail in Section 7.5.1.

7.3.3 Artificial Intelligence

artificial intelligence (AI) is often considered to be either strong AI or weak AI. Strong AI is said to be comparable or even better than human intelligence. However, there is no good definition of strong AI that is widely agreed on. The main reason is probably that *intelligence* does not have a real scientific definition. My favourite description of strong AI is from Alan Turing [91]. If a machine can pass the Turing test, then it is strong AI. In short, the Turing test is about whether or not a machine can answer arbitrary questions, such that a person (giving the questions) cannot reliably distinguish the machine from a person. So, according to Turing, if a machine answers questions like a human, then it is strong AI. None of the techniques described in this thesis, are designed to act as human beings and are therefore considered to be weak AI. Weak AI, are in general algorithms that try to simulate human or animal traits. Examples of such traits are *learning*, *reasoning*, *planning* etc. However, algorithms that make use of general natural phenomena like evolution or swarming are usually also considered to be weak AI.

Many of the algorithms used for solving combinatorial optimization problems are considered to be weak AI. This is also the case for some of the algorithms in computational biology. Three of the AI techniques described in this thesis are so-called *supervised learning* algorithms. These are artificial neural networks (Section 4.1), hidden Markov models (Section 5.2.1) and support vector regression (Section 6.2.3). All of these algorithms have in common that they are presented with a number of examples and are supposed to *learn* the general patterns in the set of examples. These algorithms differ much in how they learn from the examples and how they represent the knowledge they have learned. However, they are all capable of handling incomplete and uncertain data in the set of examples. The tabu search metaheuristic presented in this Chapter is also considered to be weak AI, simply because of the memory used to escape local minima.

Swarm Intelligence

In Section 7.5.3 we describe our approach for protein structure prediction using a search strategy borrowed from the foraging behaviour of honey bees. Such an algorithm is called *swarm intelligence* (SI) and is also considered to be AI. In nature, some animals *swarm* to achieve survival and reproductive benefits. The specific way a species of animals swarm varies, but the term is usually used to describe a group of animals (usually insects, fish or birds) that moves in the same direction and behaves similar to environmental changes. SI typically consists of a number of individuals called *agents*. Such agents are able to work independently in the environment, but usually make decisions based on communication with other agents or changes in their environment. In the case of our bee colony optimization approach, agents correspond to honey bees. A honey bee is able to collect nectar without the help of other bees, but it can also communicate with other bees (using the so-called *waggle dance*) to tell other bees about the positions of good flower beds. Another example of a swarm is an ant colony.

When using *ant colony optimization* (ACO), the agents correspond to ants who seek to find food close to their colony. Ants seek for food randomly and also leaves a so-called *pheromone trail*. This pheromone trail attracts the other ants from the colony such that trails with high pheromone contents have a higher probability of being used by the ants. The pheromone also evaporates. Long trails take a long time to travel and therefore eventually end up having lower pheromone concentration than shorter trails. ACO is suitable for solving graph problems such as the TSP [20], but it has also been applied to the protein structure prediction problem in [86].

7.4 Exact Algorithms

In 1968 Levinthal postulated that the conformational space of proteins is too large to be searched exhaustively [51]. It is therefore believed that proteins use folding pathways to reach the native structure. This is probably also one of the main reasons why many computational approaches to protein structure prediction use heuristic search algorithms to generate folding pathways. The main problems with these algorithms are that they get trapped in local minima and they cannot give a guarantee on the solution quality.

Levinthal's postulate is of course widely acknowledged. However, just because nature uses folding pathways it does not necessarily mean that computational approaches to structure prediction must use the same technique. There have been a few *exact* algorithms for protein structure prediction proposed in the literature. These algorithms implicitly search the conformational space exhaustively using advanced optimization techniques. Something that proteins in nature, of course, cannot do.

In computer science it has been known for decades that optimal solutions indeed can be found for many realistic problems, even though the solution space is very big. Some instances of problems like *vehicle routing problems* (VRP) [45], *knapsack problems* (KP) [40] and the Steiner tree problem [31, 13, 68] can be solved in reasonable time using advanced techniques from combinatorial optimization. The fact that the conformational space of proteins is astronomical large, therefore should *not* be a reason for avoiding exact algorithms.

The main advantage of using exact algorithms is that they guarantee to find the global optimum in the given model and do therefore not have the problem of getting stuck in local optimum. One of the disadvantages is that efficient exact algorithms are often much more difficult to design compared to simple search heuristics like MC and TS. The model and energy functions are therefore typically simplified such that bounds can be computed efficiently.

In this section some of the exact algorithms for protein structure prediction that exist in the literature are briefly described. We begin with an exact algorithm in the HP model, continues with the αBB algorithm and end with a description of our own exact algorithm.

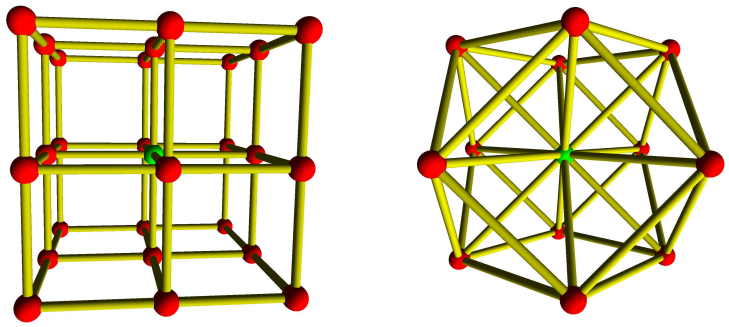


Figure 7.4: (a) The *simple cubic* (SC) lattice where each lattice node has 6 neighbours. (b) the more complex *face centered cubic* (FCC) lattice where each lattice node has 12 neighbours. The lattice vectors of the SC lattice are $(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)$, and the lattice vectors of FCC are all combinations of $(\pm 1, \pm 1, 0), (\pm 1, 0, \pm 1), (0, \pm 1, \pm 1)$.

7.4.1 Exact Structure Prediction in the HP-model

In the paper by Backofen and Will [7] an efficient constraint-based algorithm is presented. Their algorithm can solve large instances of problems in the *simple cubic* (SC) lattice and the *face-centered-cubic* (FCC) lattice (Figure 7.4). Their objective is to maximize the number hydrophobic neighbours of the string of beads. This is the same objective as for the classic HP-model, but Backofen and Will use more complex and realistic lattices.

The basic idea of their approach comes from the observation that optimal solutions often have near the maximum number of hydrophobic contacts that is possible in a lattice. So, by knowing only the number of hydrophobic amino acids, they can precompute the expected energy. Furthermore they can precompute the so-called maximally compact cores of the string of beads. This is illustrated in Figure 7.5 with a simple example. When generating a solution, all that is needed is to thread the remaining hydrophilic amino acids on the hydrophobic core. However, this might not be possible if the optimal solution does not have a maximally compact core. In that case, the algorithm iteratively threads the hydrophilic amino acids on less compact cores until it succeeds. Refer to [7] for details about computing compact cores and threading the hydrophilic amino acids on the hydrophobic cores.

Using their exact algorithm it is possible to find the global minimum structure of proteins up to 200 residues in less than a minute on a Pentium 4. Backofen and Will do not report the similarity of the global minimum structures with the native structures. This is probably because there are no similarity due to the simple energy function.

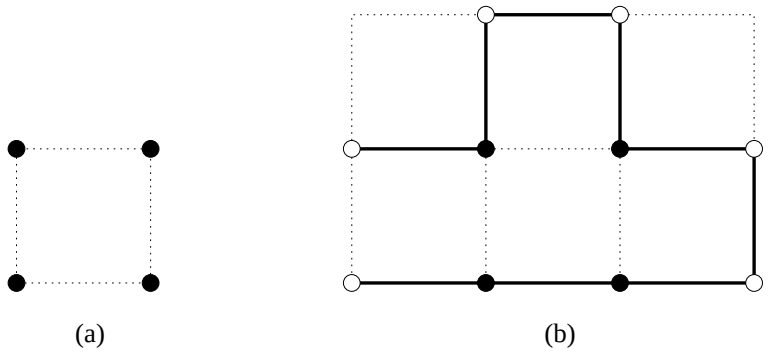


Figure 7.5: Consider the string PHPPHPPHHP with 4 hydrophobic amino acids and 6 hydrophilic amino acids. (a) shows the most compact core of 4 hydrophobic amino acids which can be precomputed. Figure (b) shows a possible threading of the string of beads to the core. When such a threading is found (if possible), the solution is optimal.

7.4.2 The αBB Algorithm

The α *Branch and Bound* (αBB) algorithm is one of the few algorithms for protein structure prediction that is based on the branch and bound paradigm. In this section, the general branch and bound paradigm is described, and the αBB by Maranas et al. [54, 4] is introduced. Our branch and bound approach for protein structure prediction is described in Section 7.5.2. Maranas et al. used the αBB algorithm to predict the global minimum energy structures of small molecules (up to 14 atoms). In [23] Eyrich et al. extended the αBB algorithm such that it can handle proteins with hundreds of atoms.

The Branch and Bound Paradigm

The branch and bound approach is an algorithm for solving various combinatorial optimization problems. It was first described in 1960 by Land et al. [44] where they used it for solving linear programming problems. Today, branch and bound algorithms are mostly used for solving NP-hard problems.

One of the basic techniques of the branch and bound paradigm is the recursive subdivision of the solution space into smaller sets. Such a recursive subdivision can also be represented by a tree, where the union of the children nodes represents the solution space of the parent node. Subdivison of the solution space is called *branching*. Branch and bound also require the computations of upper and lower bound estimates for a particular subdivision. A lower bound is a number that is equal to, or lower than, any solution value in the set. The upper bound is a value that is greater than, or equal to, the minimum solution value in the set. Obviously, a particular solution set cannot contain a global minimum solution if the lower bound is higher than an upper bound in any solution set. When such a situation occurs, the whole solution set can be disregarded (bounded) without explicitly considering each solution.

When developing a branch and bound algorithm for a particular problem,

the main goals are therefore to develop an appropriate branching scheme and an efficient lower bound algorithm. A good lower bound algorithm is therefore fast (compared to complete enumeration of the subset) and computes bounds that are close to the minimum value solutions in the sets; a so-called *tight bound*.

αBB

In [54] Maranas et al. represent molecules using a list of dihedral angles between bonded atoms. The energy function is the Lennard-Jones potential function [50]. Branching is done by considering the dihedral angle with the widest range and constructing two new subspaces corresponding to splitting the chosen dihedral angle in two separate intervals. Note that the dihedral angles here are treated as continuous variables and not discrete as for real combinatorial optimization problems.

The lower bound is computed by using theory from convex optimization theory. Maranas et al. develop a theory that shows how to compute a lower bound function, L, given the energy function V. The lower bound function, of course, has the property that it is less than or equal to the energy function for all conformations in the set. Furthermore, L is convex such that a local minimum of L is also a global minimum. In each node of the branch and bound tree, the lower bound is computed by minimizing L which can be done using standard techniques for solving convex optimization problems. The upper bound is the corresponding value of V (the conformation where L is minimum). Maranas et al. are able to find global minimum energy solutions but only for small molecules. However, Eyrich et al. [23] extend the αBB algorithm by using fixed secondary structure elements and are able to work on real sized proteins. In [23] all results are reported only for secondary structure segments derived from the native structure of the protein. As described in Chapter 4, secondary structure prediction is far from perfect and Eyrich et al. do not show how the performance of the αBB algorithm is, when the secondary structure predictions have errors.

7.4.3 Protein Threading

Another large category of tertiary structure prediction algorithms are the *threading* based algorithms. In a typical threading algorithm, a set of structures representing different folds are known. Using the threading algorithm, the most likely fold of an amino acid sequence can be identified and used as a template for later model building.

This task is accomplished by a *threading* of the amino acids of the protein with unknown structure on each of the fold structures. A threading is found by aligning the amino acid sequence to a fold structure, possibly using gaps and insertions, as illustrated in Figure 7.6. Each possible threading has an energy and the threading algorithm finds the threading with minimum energy. Such a minimum energy threading is found for each of the fold structures and the threading with minimum energy over all fold structures is assumed to be the fold structure of the amino acid sequence with unknown structure. The number of possible threadings on a single fold structure is exponential and in [46], the

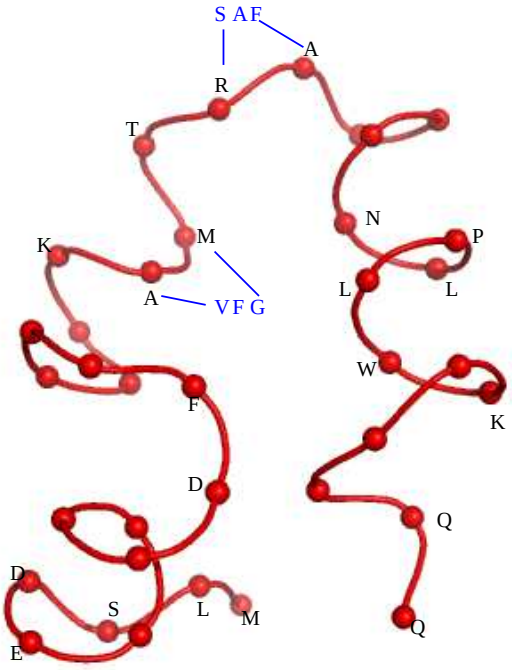


Figure 7.6: The sequence *MLSDEDFKAVFGMTRSAFANLPLWKQQ* is aligned to the fold structure. The blue residues correspond to insertions. The residues with no letters correspond to deletions.

problem is shown to be NP-hard for energy functions with a sum pairwise terms.

7.4.4 Example of an Exact Threading Algorithm

One possible threading model is described in [47]. Here, a so-called *core structural model* (CSM) is made for all fold structures. A CSM consists of a sequence of loops and secondary structure elements. The secondary structure elements have fixed length and the loop regions have a minimum and maximum length. A valid threading of an amino acid sequence is therefore an assignment of subsequences to each loop region and secondary structure element such that the intervals are satisfied (Figure 7.7). The energy function both consists of positions of single amino acids and pairwise terms:

$$f(T) = \sum_i g_1(i, t_i) + \sum_i \sum_{j>i} g_2(i, j, t_i, t_j)$$

Where g_1 is a function depending on the core segment i located at the t_i'th amino acid in the sequence. Likewise g_2 is a function of pairs of core segments

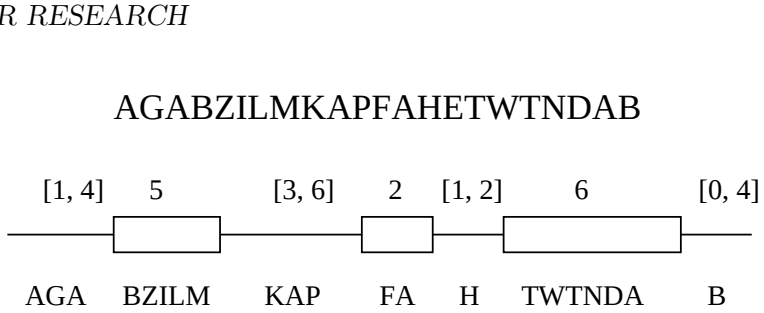


Figure 7.7: An example of a core structural model. The lines correspond to variable length loop regions and the boxes correspond to secondary structure elements. The upper intervals correspond to the number of residues allowed in the regions and the letters show an example of a valid threading of sequence $AGABZILMKAPFAHTWTNDAB$.

and their positions in the sequence. The actual values of the energy could depend on secondary structure prediction, amino acid burial, hydrophobicity, statistical derived potentials, etc. The problem of finding the threading with global minimum energy is NP-hard because of the pairwise terms and in [47] the problem is solved using the branch and bound technique.

7.5 Our Research

We have developed three different approaches for reconstructing C_α-traces using techniques from combinatorial optimization. Our first approach described in Section 7.5.1 applies exact values of half-sphere-exposure (HSE) to reconstruct C_α-traces using Monte Carlo search and Tabu search. In Section 7.5.2 we describe our exact approach using a branch-and-bound technique for decoy generation. In this approach we also apply predicted measures and the algorithm can therefore be considered as *de novo*. In Section 7.5.3 we describe our artificial intelligence approach. It is based on a swarm intelligence which mimics the foraging behaviour of honey bees. All of these approaches are briefly described here. For more details refer to the corresponding papers.

7.5.1 Paper: Reconstructing Protein Structure from Solvent Exposure using Tabu Search

In our paper *Reconstructing Protein Structure from Solvent Exposure using Tabu Search* [63] we compare the performance of *Monte Carlo* (MC) search and *tabu search* (TS) on a simple protein structure prediction problem. In addition to the MC and TS comparison we also estimate the information contents of the newly introduced *half-sphere-exposure* (HSE) measure [29]. Figure 7.8 contains a short description of HSE from the paper.

Determining the information contents of the HSE measure or the CN measure is important. Both measures can be predicted from amino acid sequence with reasonable accuracy and can therefore be used for de novo prediction. However, in the paper described here we only use exact measures computed from the native structure of the protein.

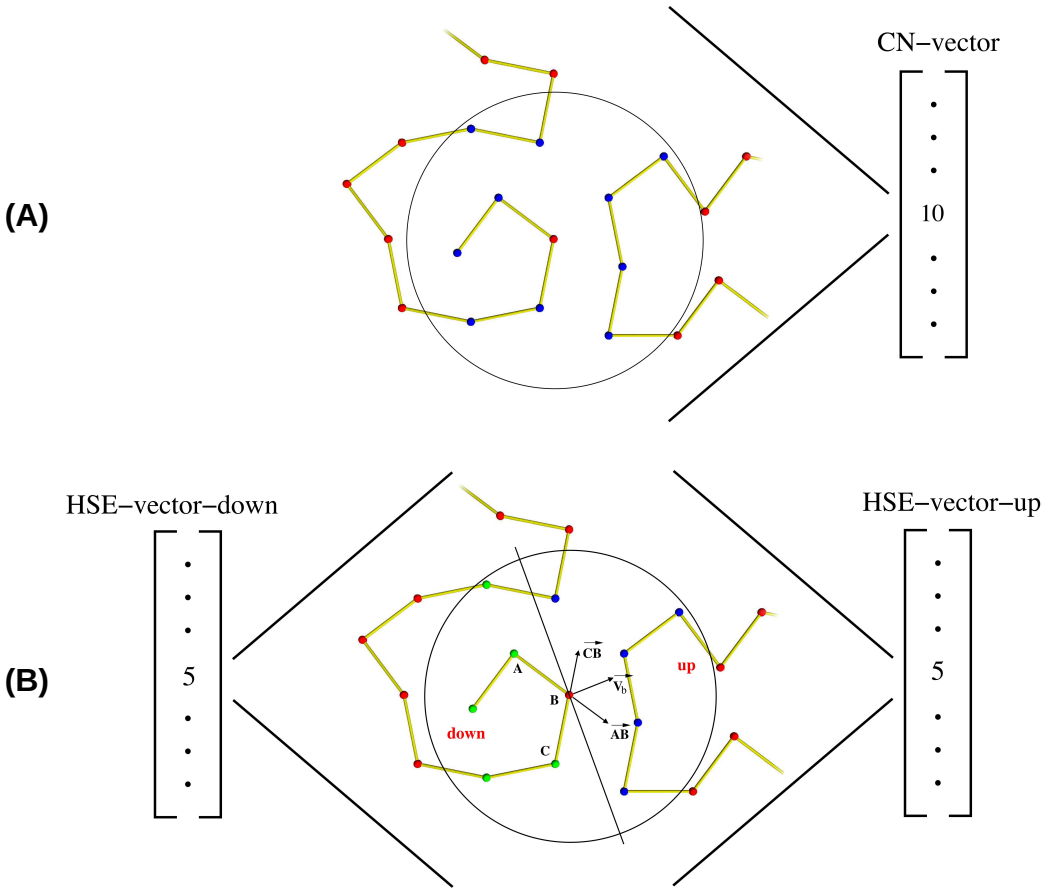


Figure 7.8: *(From [63]).* The extent to which an amino acid in a protein is accessible to the surrounding solvent is highly dependent on the type of amino acid. In general, hydrophilic amino acids tend to be near the solvent accessible surface, while hydrophobic amino acids tend to be buried in the core of the protein. To measure this effect, several solvent exposure measures have been proposed [49, 27, 17, 15, 73, 74, 75], and one of these is the contact number measure (CN) [75]. The CN of a residue is the number of C_α atoms in a sphere centered at the C_α atom of the residue in question (Figure A). The CN of all residues of a protein is called the CN vector. The CN vector is well conserved and can be predicted with high accuracy [41]. While the CN measure uses a single sphere centered at the C_α-atom, the HSE measure considers two hemispheres. Two values, an up and a down value, are associated with each residue, corresponding to the upper and lower hemisphere. The geometry of the HSE construction is shown schematically in Figure B. Given the positions of 3 consecutive C_α atoms (A, B, C), the approximate side-chain direction $\vec{V_b}$ can be computed as the sum of $\vec{AB}$ and $\vec{CB}$. The plane perpendicular to $\vec{V_b}$ cuts the sphere centered at B in an upper and a lower hemisphere. The up and down HSE values measure two fundamentally different environments of an amino acid, one of them corresponding to the neighbourhood of the side chain [29]. The HSE measure compares favorably with other solvent exposure measures in terms of computational complexity, sensitivity, correlation with the stability of mutants and conservation. An important advantage of the HSE measure is that it can be calculated from C_α-only or other simplified protein models. Therefore, it forms an attractive alternative to the use of the CN measure in protein structure prediction methods [87].

Our strategy for evaluating the information contents of HSE and comparison of the algorithms is the following. A random C_α-trace is constructed and iteratively improved by either MC or TS such that the HSE vector of the C_α-trace is similar or close to the HSE-vector of the real protein. This is done by minimizing the energy function

$$\mathrm{E}(A,B) = \sqrt{\frac{\sum_{i=1}^{N}\left((A_{u_i}-B_{u_i})^2+(A_{d_i}-B_{d_i})^2\right)}{2N}},$$

where $\{A,B\}_{u_i}$ and $\{A,B\}_{d_i}$ are the up and down values of the i'th index of the HSE-vectors. N is the length of the vectors. $\mathrm{E}(A,B)$ is the energy of structure S_A where A is the HSE vector of S_A and B is the HSE vector derived from the native structure. These definitions are easily specialized to the CN measure.

To discretize the conformational space of the C_α-trace, the C_α-atoms are confined to be positioned on a lattice. In addition to the simple 2D quadratic lattices and 3D cubic lattices already discussed in Figure 7.1, we also consider the more complex lattices FCC (Figure 7.4(b)) and *high coordination* (HC) lattice (Figure 7.9). A structure is represented by a list of directions in the lattice for all but the first C_α-atom. The move set used by both algorithms, MC and TS, consists of all possible changes of up to three consecutive directions. Using this terminology, the simple move set described in Figure 7.2 contains respectively 2 changes (A), 3 changes (B) and 1 change (C). In addition to this move set, we also allow one index change at a non-endpoint. This results in a translation of the string of beads after the index change.

The MC algorithm is implemented as described in section 7.3.1 and the TS algorithm uses a new tabu definition. The trivial tabu definition is to store previously visited solutions in a tabu list and then prevent the algorithm from visiting them. When using this tabu definition, our experience is that it takes very long time to escape local minima. This is mainly because the tabu list needs to be filled with all structures in a neighbourhood around a local minimum before it can escape the local minimum. When using complex lattices, there are many different structures that are almost structural equal and the tabu list therefore becomes very large before the local minimum is escaped. This is a problem we reduce by defining the concept of explicit- and implicit tabu structures (see Figure 7.10).

In paper [63] there are three experiments which are briefly described here.

a. The first experiment determines suitable values of the tabu difference (ϵ) and the tabu list size. This is done by running the TS algorithm on 20 initially random structures for a small peptide and optimize each structure until a zero-energy structure is found or a maximum of 15 minutes have passed. An average energy of the 20 final structures is computed and plotted in Figure 7.11. When the tabu difference is zero, the TS algorithm behaves as a regular TS algorithm where only previously visited structures are tabu. The figure therefore shows, that our use of implicit tabu structures improves the performance of the TS algorithm considerably on this experiment. On the other hand, one should be careful that

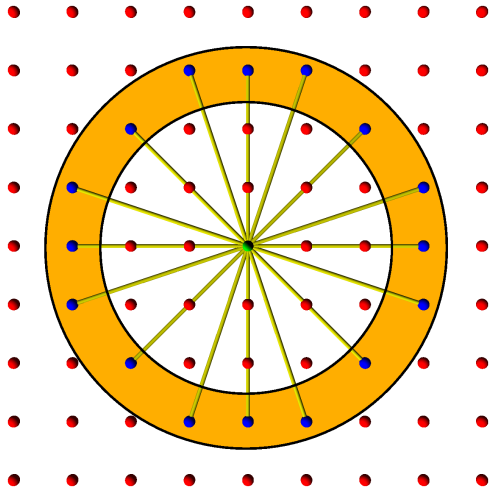


Figure 7.9: This is a 2D illustration of the *High coordination* (HC) lattice. A high coordination lattice has an underlying cubic lattice with unit length less than $3.8/N$ Å for some integer $N > 1$. Cubic lattice points are connected in the high coordination lattice if their Euclidean distance is between $3.8 \pm \beta$ for some $\beta > 0$. The high coordination lattices used in our experiments are named HC4 and HC8 corresponding to their N value (4 and 8). The β value is 0.2 for all applied HC lattices. The figure shows a 2D high coordination lattice with $N = 3$ and $\beta = 0.4$.

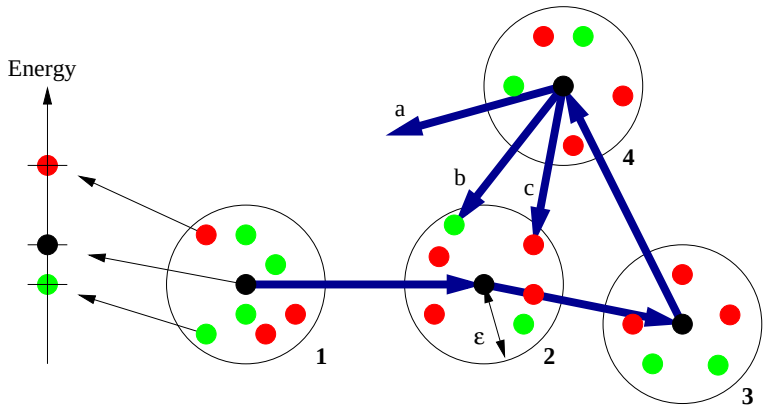


Figure 7.10: *(From [63]).* We keep a list of previously visited structures in a so-called explicit tabu list. Each structure in the explicit tabu list defines a set of implicit tabu structures. Given a structure E in the explicit tabu list, a structure I is said to be implicit tabu if the distance-RMSD (dRMSD) between E and I is less than ϵ and the energy of I is greater than or equal to the energy of E. The adjustable parameter ϵ is called the tabu difference. The figure illustrates a sequence of visited structures (black points) in a solution space. Only the visited structures are inserted in the explicit tabu list. The additional green and red points correspond to structures within ϵ dRMSD of the explicit tabu structures. Green points are structures with lower energy and red points are structures with higher energy than the explicit tabu structure. When choosing a new solution in the neighbourhood three things can happen (as illustrated in the figure), a) A solution is more than ϵ dRMSD away from all explicit tabu structure. b) the solution is within ϵ dRMSD, and the energy is lower than the explicit tabu structure, c) the solution is within ϵ dRMSD, and the energy is higher than the explicit tabu structure. Structures that comply with case c are said to be implicit tabu and cannot be visited. Note that when $\epsilon = 0$ the search heuristic works as a regular TS heuristic since only visited structures become tabu. The use of implicit tabu structures is new in the context of protein structure prediction.

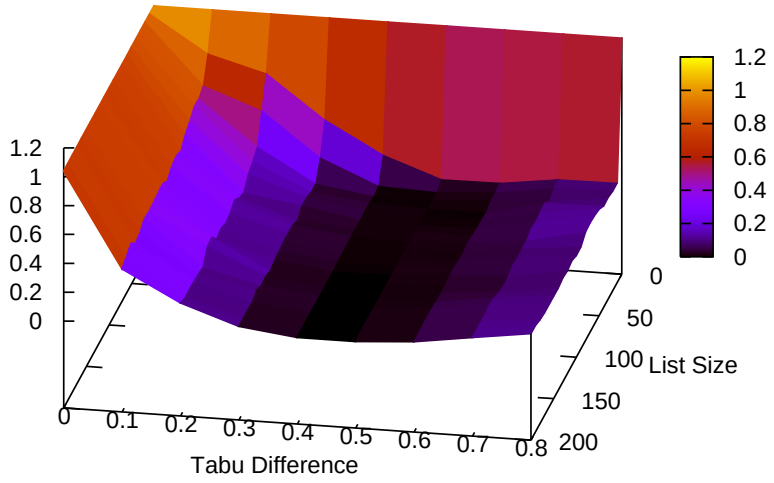


Figure 7.11: Average energy for different combinations of tabu difference and tabu list size.

the tabu difference does not become too large and consequently marks good solutions tabu. In the example shown in the figure, the best value of the tabu difference is between 0.4 and 0.5.

b. The second experiment compares the performance of TS and MC. This is done using the same test framework as in experiment *a* but with different lattices. The results are shown in Figure 7.12. In our paper [63] the y-axis is linear, however, the logarithmic y-axis used here more clearly describes the performance of both algorithms.

c. The purpose of the third experiment is to evaluate the information contents of the CN measure and HSE measure. This is done by using the TS algorithm, such that a number of structures with minimum energy is found. These structures with low energy are then compared to the native structure of the protein in terms of RMSD. In [63] this is done for 5 small proteins. For some of the proteins, we find many different structures with zero or near zero energy which indicates that the energy landscape has many global and local minima.

Based on the experiments, we conclude that the use of implicit tabu structures can increase the performance of TS based search algorithms. Our results also show that TS has a better performance than a typical MC algorithm on this type of problem. However, it is important to note that there are many alternative versions of MC that we have not tested. The HSE vs. CN experiments clearly show that the information contents of HSE is higher than CN. In the paper we also show that the approximate directions of the side-chains are much closer to the native structure for the HSE-optimized structures.

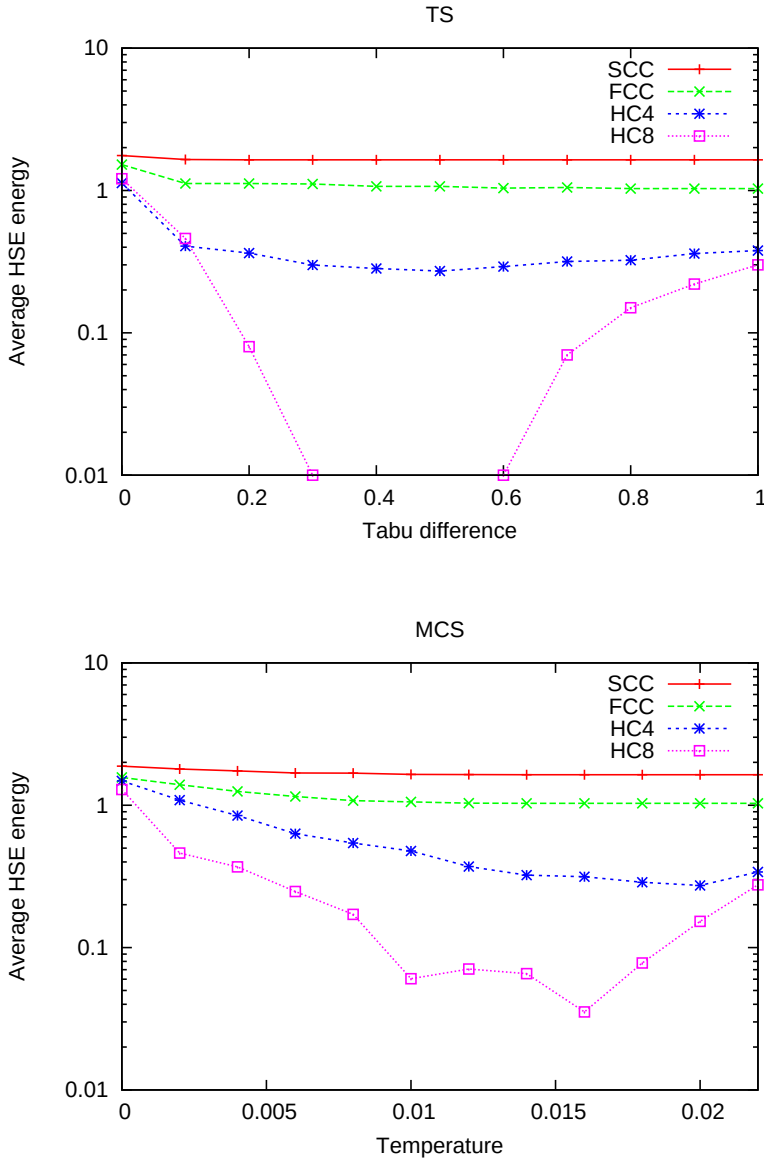


Figure 7.12: Average energy using lattices of different complexity vs. adjustable algorithm parameters. The upper plot shows the average performance of the TS algorithm for a range of tabu differences. For tabu differences 0.4 and 0.5 the average HSE energy is 0 (not shown in a the logarithmic plot). The lower plot shows the performance of the MC algorithm for a range of different temperatures. There are no temperature for the MC algorithm that gives the same performance as the TS algorithm when the tabu difference is between 0.4 and 0.5.

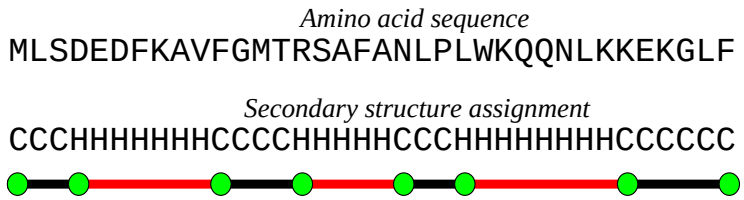


Figure 7.13: The secondary structure is predicted from the amino acid sequence and used for creating segments.

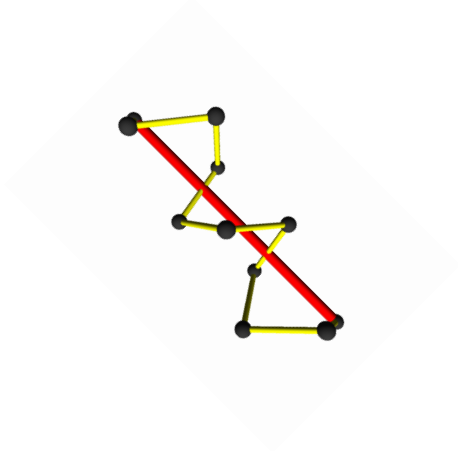


Figure 7.14: A coil segment with valid positions of C_α-atoms.

7.5.2 Paper: Protein Decoy Generation using Branch and Bound with Efficient Bounding

As described in the previous section, the HSE/CN-based energy function has many local minima. Furthermore, the information contained in the HSE or CN measure is not enough to accurately reconstruct C_α-traces of large proteins. In the study described here, we attack these problems by using an exact algorithm that guarantees to find structures with global minimum energy. We also add more predictable information in the form of secondary structure classifications and predicted compactness (radius of gyration).

The Model

The basic idea in our exact approach [66, 67, 65] is to reduce the complexity of the model as illustrated in Figure 7.13. First, the secondary structure of the protein is predicted using PSIPRED [57], then this prediction is used for creating the segments of secondary structure. The purpose of a segment is to define an approximate path in space for the amino acids that it represents. This is done by positioning the first and last amino acids of the segment at the two end points of the segment. Figure 7.14 shows an illustrations of a helix segment together with valid positions of the C_α-atoms of the amino acids.

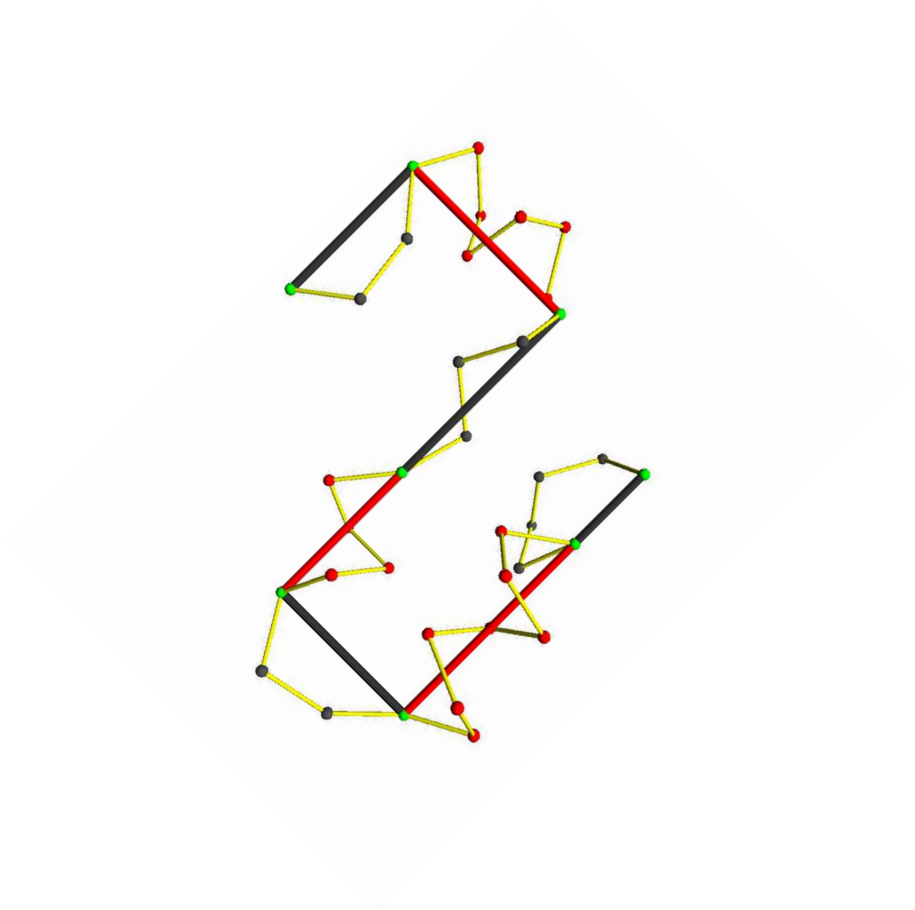


Figure 7.15: An example of a complete structure. In this example two types of secondary structure is represented. The red segments correspond to helices and the black segments correspond to coils.

We discretize our model by allowing these segments to only have a set of predefined directions. There is a trade-off between the number of allowed directions and computational tractability. Our ad-hoc experiments show that we can solve problems in reasonable time when using the 12 uniformly distributed directions from the FCC lattice as described in Figure 7.4(b). In addition to the discretization of the allowed directions of the segments, we also allow a limited set of valid positions of C_α-atoms of a segment. These are called *segment structures*. For helices and sheets, we generate u such segment structures by rotating one structure (having perfect geometry) around the axis defined by the segment. For coil segments, we query a library of coil fragments, to find the most similar sequences and use them as segment structures. A structure represented by this model is called a *complete structure* and is illustrated by the example in Figure 7.15. Refer to [66] for more details about the model and generation of segment structures.

Solving the Problem

Given an amino acid sequence with m segments and u possible segment structures for each segment, the total number of complete structures, N, allowed by this model is

$$N = 4 \times 11^{m-2} \times u^m \tag{7.1}$$

In the equation above, symmetric structures are not counted twice. The total number of complete structures is exponential and too high for a complete enumeration even for small proteins. On the other hand, we have designed a model with many geometric constraints which allows us to compute lower bounds in the branch and bound paradigm efficiently. Our branch and bound algorithm is called: *Efficient Branch and Bound Algorithm* (EBBA) throughout this text. A node in the branch and bound tree corresponds to a partial solution where one or more directions of the segments have been fixed and zero or more segment structures have been fixed. Even though such a partial solution represents a high number of structures, they are heavily geometric constrained. Lower bounds can be computed by taking advantage of these geometric constraints as described in [66, 67].

Experiments and Results

We have tested EBBA on 6 proteins. In all cases we find structures with similar CN and HSE vectors compared with the predicted CN and HSE vectors (example in Figure 7.16). However, even though the CN and HSE vectors matches to some extend, the corresponding structures are not always similar. In other words; the lowest energy structures are in many cases different from the native structures. EBBA is therefore modified such that it returns the 10.000 global minimum energy structures and we show that in this set, good decoys exist for all proteins in our benchmark. EBBA should therefore not directly be used for protein structure prediction, but it is a successful decoy generator compared with other decoy generators in the literature [67, 66].

7.5.3 Paper: Protein Structure Prediction using Bee Colony Optimization Metaheuristic

Our latest approach for protein structure prediction is inspired by swarms of honey bees. This research is work in progress, but contains important results in the context of this study and is therefore included here. The draft of the paper is in Appendix E page 165.

In nature, honey bees collect nectar to produce honey in the hive. Honey is the main food source for the bees, so an important task in a bees life is to collect as much nectar as possible. Nectar is produced by flowers in limited amounts. The perfect place for a nectar collecting bee therefore is a flower field. Since bees do not have a map of flower fields in the neighbourhood of their hive, evolution has provided them with a search strategy to maximize the collection of nectar. Before describing this strategy, notice the similarity with the protein structure

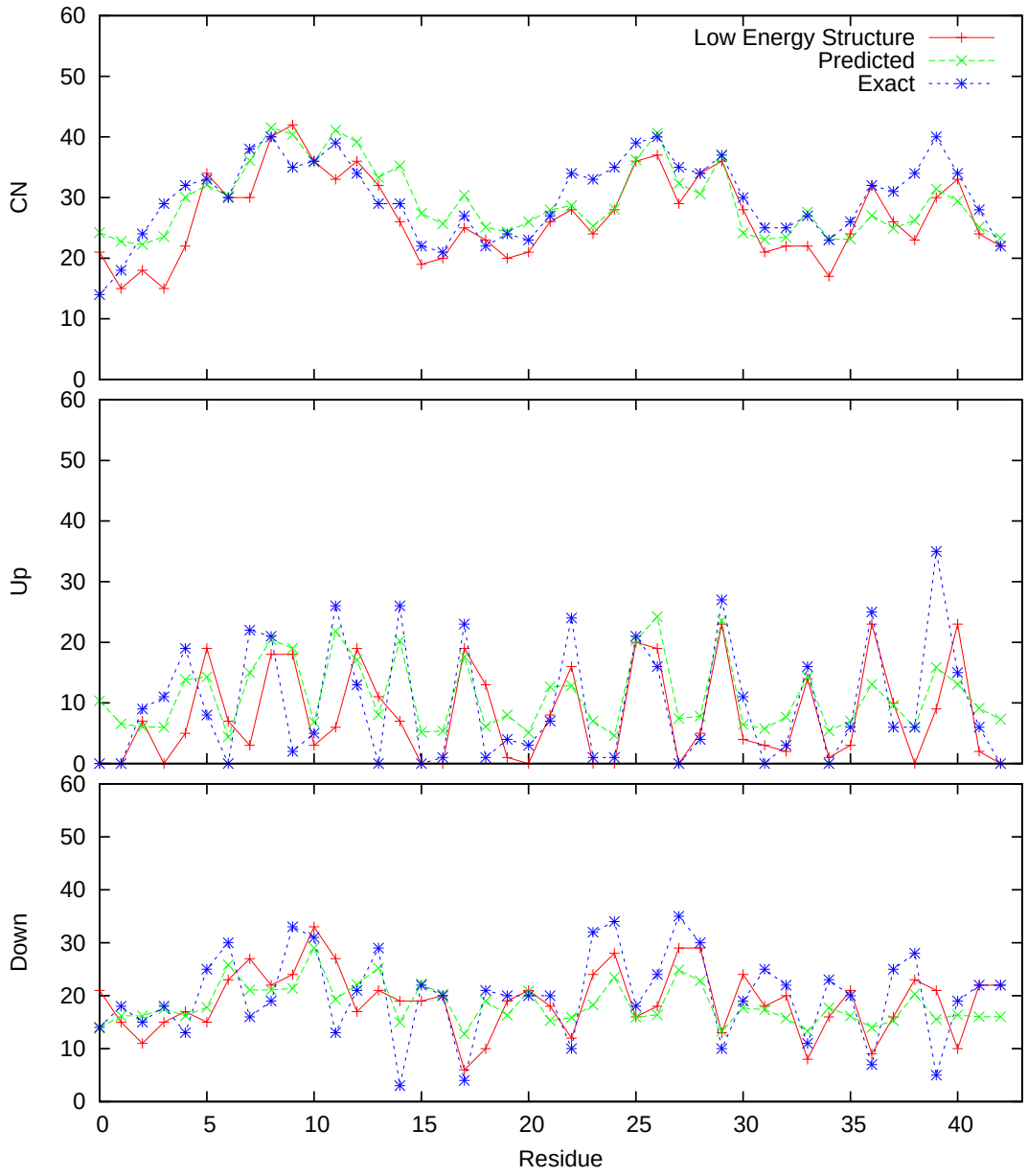


Figure 7.16: An example of CN, up and down vectors for the protein 1FC2. The *low energy structure* is found by EBBA. The *predicted* vector is the prediction from LAKI [92]. The *exact* vector is derived from the native structure and only used for evaluation.

prediction problem or other optimization problems in general. As illustrated in Figure 2.8 page 21, protein structure prediction is about finding the global minimum structure in an energy landscape. Honey bees do not know where the good flower beds are, and we do not know where the minimum energy structures in the conformational landscape are. A simple idea is therefore to apply the bees search strategy to the protein structure prediction problem.

The use of the foraging behaviour of honey bees in combinatorial optimization problems was proposed simultaneously in 2005 by Pham et al. [71] and Karaboga et al. [33]. They later published a number of applications and results of the so-called *bee colony optimization* (BCO) algorithm [72, 36, 35, 34]. In [24] we present our approach for protein structure prediction using BCO. The idea of using BCO for protein structure prediction is not entirely new. In [8] Bahamish et al. used BCO for finding the native state of the 5-residue peptide *met-enkalphin*. The native structure of small polypeptide-chains having only 5 residues is usually not considered to be difficult to predict. Our BCO algorithm is therefore the first algorithm in literature that can handle real-sized proteins (proteins up to 136 residues are considered).

The Bees' Strategy

The basic bee strategy is to organize the swarm such that *many* bees are deployed at high quality flower beds with much nectar and *few* bees search for new flower beds or harvest nectar from low quality flower beds. To accomplish these tasks, the strategies of a bee can roughly be divided in three categories: scouts, workers and onlookers. Here we briefly describe their task in nature and how they correspond to search operations in our BCO algorithm.

- *A **scout** bee in nature:*
  It flies in random directions and eventually finds a flower bed. It collects nectar from the nearby flowers and returns to the hive. In the hive, it performs a so-called *waggle dance* that communicates the amount of nectar in the flower bed and the position of the flower bed to the other bees.

- *A **scout** bee in the algorithm:*
  It corresponds to a valid structure constructed randomly. The energy of the random structure corresponds to the amount of nectar in the flowerbed. The waggle dance corresponds to evaluating the energy and storing the structure in a data structure.

- *An **onlooker** bee in nature:*
  It watches the waggle dances performed by either scout bees or worker bees and decides to collect nectar in a flower bed as shown by one of the waggle dances. If the waggle dance indicates a high quality flower bed, the chance of selecting that particular flower bed is higher.

- *An **onlooker** bee in the algorithm:*
  It is first assigned to an existing protein structure (corresponding to a worker bee or scout bee). This assignment depends on the energy of the

structure such that the chance of being assigned to a low energy structure is higher. Then the onlooker bee is deployed at a structure in the neighbourhood of the assigned structure (possibly found by local search heuristic started at the assigned structure). If the neighbourhood structure is better than the assigned structure, the onlooker replaces the worker bee and becomes a worker bee. Otherwise it flies back to the hive.

- *A **worker** bee in nature:*
  Worker bees are like onlooker bees except that they do not consider other bees waggle-dances. Instead they just fly back to their old flower beds to collect more nectar. If the nectar in their flower bed depletes, they are redeployed as scouts.

- *A **worker** bee in the algorithm:*
  It corresponds to a solution in the conformational space. If the solution is improved (by an onlooker bee) the worker bee is redeployed as either a scout bee or onlooker bee. Otherwise, the worker bee represents the same solution. If some onlooker bee has not improved the site of a worker bee for a pre-specified number of iterations, the site of the worker bee is said to be *exhausted*. In that case, the worker bee is redeployed as a scout bee.

We do not consider scout bees, worker bees and onlooker bees as individual objects in our algorithm. We use the concepts described above to maintain a set of so-called working sites and onlooker sites. This is illustrated in Algorithm 1.

Algorithm 1: BEE-COLONY-OPTIMIZATION

input : S, W, O, $StopT$, $Exhaust$
output: A low energy structure

1 Create $S + W$ working sites by random *(corresponds to deploying S scouts and W worker bees. Worker bees are initially deployed randombly like scouts)*
2 **while** *Stopping criterion is not met* **do**
3 Promote O sites as onlooker sites *(Assign onlooker bees to flower beds using onlooker selection strategy)*
4 **for** *Each onlooker site* **do**
5 Find a neighbourhood site *(using onlooker site improving strategy)*
6 If the neighbourhood site is better than the onlooker site, move the onlooker site to the neighbourhood site. *(corresponds to redeploying the onlooker bee as a worker bee and sending the old worker bee back to the hive*
7 **end**
8 Make all onlooker sites working sites
9 Abandon the S worst working sites and create S new working sites *(using the scout bee strategy)*
10 If a working site has not been improved by an onlooker bee in $Exhaust$ iterations, abandon the working site and construct a new working site *(Corresponds to depletion of nectar and redeployment of worker bees as scout bees)*
11 **end**
12 **return** The best observed working site

Note that step 6 in the algorithm above, might not necessarily be hill climbing which is used here. It would be interesting to test the performance of a strategy where worse solutions can be selected with some probability (i.e. using the Monte Carlo acceptance criteria).

Experiments and Results

We have made experiments where the BCO algorithm is run on the same model as used in EBBA (Section 7.5.2). BCO often finds the optimal solution faster than EBBA - but not always. There are also examples where BCO does not find optimal structures in the 48 hours time limit (optimal solutions were found by EBBA in less than 48 hours for the proteins tested). When using such low complexity models, that can be solved to optimality in reasonable time, EBBA is therefore the preferred algorithm. However, EBBA cannot solve large problems in reasonable time (in terms of model complexity and protein length). If we use a higher complexity model by increasing the allowed directions and rotations, the BCO algorithm is able to find structures with better energy than EBBA.

We have also compared the BCO algorithm with a simple simulated annealing algorithm which uses the same move set as BCO and the cooling scheme is chosen such that it spends the same amount of time as the BCO algorithm (48 hours). The results show that BCO outperform SA by finding structures with

lower energy than BCO. Refer to [24] Appendix E, page 165 for a table of the results.

7.6 Chapter Summary

When treating the protein structure prediction problem as a combinatorial optimization problem, we typically need to discretize the conformations of the polypeptide chain. Different discretizations have been proposed in the literature; some use a predefined set of allowed angles of the ϕ and ψ angles. Others, confine the C_α-atoms to be positioned on a lattice. Even though discretization gives a reduced and finite number of possible structures, it is often not feasible to find the minimum energy structure using complete enumeration because of the exponential number of structures. It has also been shown that even one of the simplest formulations of the protein structure prediction problem (the HP-model) is NP-hard. In this chapter we show examples of heuristic and exact algorithms for solving various formulations of the protein structure prediction problem. We also briefly introduce our own algorithms for protein structure prediction that are based on techniques from combinatorial optimization.

Chapter 8

Conclusions and Future Directions

The protein structure prediction problem remains a very difficult problem to solve. There has been some progress and successes in the field. However, these are mainly for proteins with homologue counterparts in PDB. While homology based algorithms usually improve when the PDB grows, it is generally not the case for *de novo* prediction algorithms. In my oppinion, structure prediction of template *free* proteins is much more interesting and intellectual challenging than structure prediction of template based proteins. If we want to predict the structure of template free proteins, we have to learn the real mechanisms behind protein folding.

While it is not possible to state exactly why the protein structure prediction problem is so difficult, we know of at least two sub problems that must be solved. One sub problem is to find a computational tractable energy function that approximates the natural energy reasonably well. The other problem is to develop a search algorithm that finds the low energy structures in a huge conformational space. Here, these sub problems are stated as being two seperate problems. However, it is quite possible that they are very intertwined. In the exact algorithm developed during this study, we are able to implicitly search the whole conformational space. However, this can not be done for arbitrary energy functions, so in this case, the energy function and the search algorithm cannot be considered seperately.

I am convinced that the protein structure prediction problem will be solved, such that the native state of *any* amino acids sequence can be predicted with high accuracy. This will probably take some decades of research and perhaps require new computational paradigms. While there are some progresses from year to year, it is not clear if the solution to the protein structure prediction problem will come from many small improvements or one revolutionary idea. I therefore think that it is important to support high risk science in this field, such that untraditional approaches can be developed and tested.

8.1 Main Contributions

The main contributions to the field of protein structure prediction and model quality assessment in this study, in a non-prioritized order, are:

- We show that the half-sphere-exposure measure is more information rich compared to the traditional contact number measure [63].
- We have proposed a new tabu definition and we show that it gives a better performing search algorithm compared to the traditional tabu definition [63].
- We have proposed a discrete and flexible model for representing C_α-traces [66, 67].
- We have developed a branch and bound algorithm (EBBA) that is able to find the lowest energy solutions in this discrete model in reasonable time [66, 67]. This is mainly because of the efficient computation of lower bounds.
- We have developed a heuristic algorithm based on the foraging behaviour of bees to find low energy structures in the discrete model [24].
- We show how to extract distance constraints from alignments and use them for model quality assessment [64].
- We show how to select a good subset of those distance constraints using information from distributions of contact number probabilities [64].

8.2 Future Directions

The future directions of this study are many and only a few of the most promising ideas are listed here.

- Even though EBBA is a *de novo* algorithm, it could be interesting to make use of techniques from homology modeling. One simple idea is to detect the best template in PDB and fix the conserved segments. EBBA should therefore only work on the unconserved parts which eventually results in a much simpler problem. Another, more flexible, way of using homology techniques is to use the distance constraints found by our MQA algorithm.
- Improvements of the energy function of EBBA could improve the quality of the global minimum structures considerably. It would therefore be interesting to use an additional physics based energy function. However, it is not trivial to compute tight lower bounds for energy functions based on residue pairwise functions, so this would require more research.
- The optimization algorithm for selecting a good subset of the distance constraints could be improved considerably. The current algorithm is based

on a greedy approach that finds a local optimum. We should test if improving the optimization algorithm eventually would improve the MQA algorithm.

In my opinion, the most promising idea is the improvements of EBBA. Exact algorithms for protein structure prediction have received very little attention in the literature. This is probably because they are more difficult to develop than heuristic algorithms. However, we basically show that it *is* possible to implicitly sample the whole conformational space with a proper discretization and therefore attack the second major problem described in Chapter 3. A more detailed energy function that allows for tight lower bound computations is therefore on top of my wish list.